Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FINAL DE CARRERA

TÍTULO: Representación y edición de un plan de vuelo sobre un modelo digital de elevaciones en entorno .NET

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Sistemas de Telecomunicación

AUTOR: David Lorente Palacín

DIRECTORES: Cristina Barrado Muxí y Juan López Rubio

FECHA: 7 de septiembre de 2007

Título: Representación y edición de un plan de vuelo sobre un modelo digital de elevaciones en entorno .NET

Autor: David Lorente Palacín

Directores: Cristina Barrado Muxí y Juan López Rubio

Fecha: 7 de septiembre de 2007

Resumen

El presente trabajo utiliza la tecnología .NET y el lenguaje de programación C# para desarrollar una aplicación capaz de visualizar y editar un plan de vuelo para un avión no tripulado y mostrar los datos orográficos de la zona a sobrevolar.

La aplicación creada es capaz de visualizar un modelo digital de elevaciones que dispone de los datos de toda la extensión del territorio de Catalunya.

El modelo del plan de vuelo se ha implementado siguiendo los conceptos basados en el sistema de navegación de área (RNAV) con los que trabaja el grupo de investigación ICARUS.

Puesto que el plan de vuelo debe ser especificado en coordenadas geográficas (espacio tridimensional) y el editor trabaja en una pantalla (espacio bidimensional) también se recogen en el presente trabajo la descripción de los distintos sistemas de coordenadas utilizados así como la proyección entre los mismos.

La interfaz de usuario permite a un operador editar un plan de vuelo de una manera gráfica e intuitiva, establecer las relaciones entre las distintas entidades que componen un plan de vuelo y verificar su correcta definición.

Del uso de la aplicación y la edición del plan de vuelo resulta un fichero en formato XML que contiene la información necesaria para reproducir el plan de vuelo a bordo del UAV. Este fichero también puede ser usado como entrada para futuras aplicaciones.

El resultado obtenido consiste en una aplicación Windows estándar que puede ser utilizada para editar satisfactoriamente un plan de vuelo.

Title: Flight plan representation and edition on a digital terrain model with .NET

Author: David Lorente Palacín

Directors: Cristina Barrado Muxí y Juan López Rubio

Date: September, 7th 2007

Overview

This project uses .NET technology and C# programming language to develop an application able to visualize and publish a flight plan for unmanned airplane and show orographic data of the zone to fly over.

Devoloped application is able to visualize a digital model of elevations that contains the data of all Catalonia region.

The flight plan model has been implemented following the concepts based on area navigation method (RNAV) which ICARUS investigation group works with.

Cause the flight plan must be specified in geographical coordinates (three-dimensional space) and the editor works over a screen (bidimensional space), the present document also contains the description of the different coordinates systems as well as the projection between them.

The user interface allows an operator to edit a flight plan in a graphical and intuitive way, establish the relations between the different entities that compose a flight plan and verify its correct definition.

As a result of edit a flight plan the application is a file in XML format that contains the information needed to reprocude the flight plan on board. This file also can be used like input for future applications.

The final result consists of an Windows standard application that can be used to edit a flight plan properly.

“All children, except one, grow up.”

A mi madre.

ÍNDICE

INTRODUCCIÓN

Este trabajo se enmarca dentro de la línea de investigación del grupo ICARUS, en el cual se desarrollan sistemas que se embarcan dentro de un UAV.

Un UAV (Unmanned Aerial Vehicle) es un avión no tripulado que combina diferentes mecanismos encargados de pilotarlo. El UAV tiene diversas aplicaciones, normalmente en el ámbito militar; pero cada vez más se están desarrollando nuevas aplicaciones en el civil, dentro de las que se encuentra el grupo de investigación ICARUS.

El objetivo del grupo es desarrollar nuevas tecnologías a bordo de un UAV orientadas a la detección, control y análisis de incendios forestales. Los datos adquiridos en tiempo real a través del UAV deberán permitir optimizar las estrategias para la extinción del fuego a quien tome dichas decisiones.

El presente proyecto se centra en la fase previa al vuelo y utiliza la tecnología .NET para desarrollar una aplicación que permita a un operador de tierra editar y verificar el plan de vuelo de un UAV de una manera fácil e intuitiva. En un futuro la misma interfaz gráfica puede usarse también como estación de control de tierra durante el vuelo.

Para ayudar en el proceso de diseño el sistema dispone del modelo digital del terreno de la zona que se va a sobrevolar con las alturas de los posibles obstáculos. El operador debe ser capaz de reutilizar fragmentos de un plan de vuelo para la generación de otros planes. Además el plan de vuelo debe ser verificado en cada una de sus etapas.

La aplicación debe permitir generar un documento en formato XML que contenga la información del plan de vuelo editado para que éste pueda ser leído por otras aplicaciones o cargado directamente en el UAV.

La configuración que aparece en la figura I.1 es el escenario inicial en el que se enmarca la aplicación donde un operador de tierra diseña un plan de vuelo con el editor.

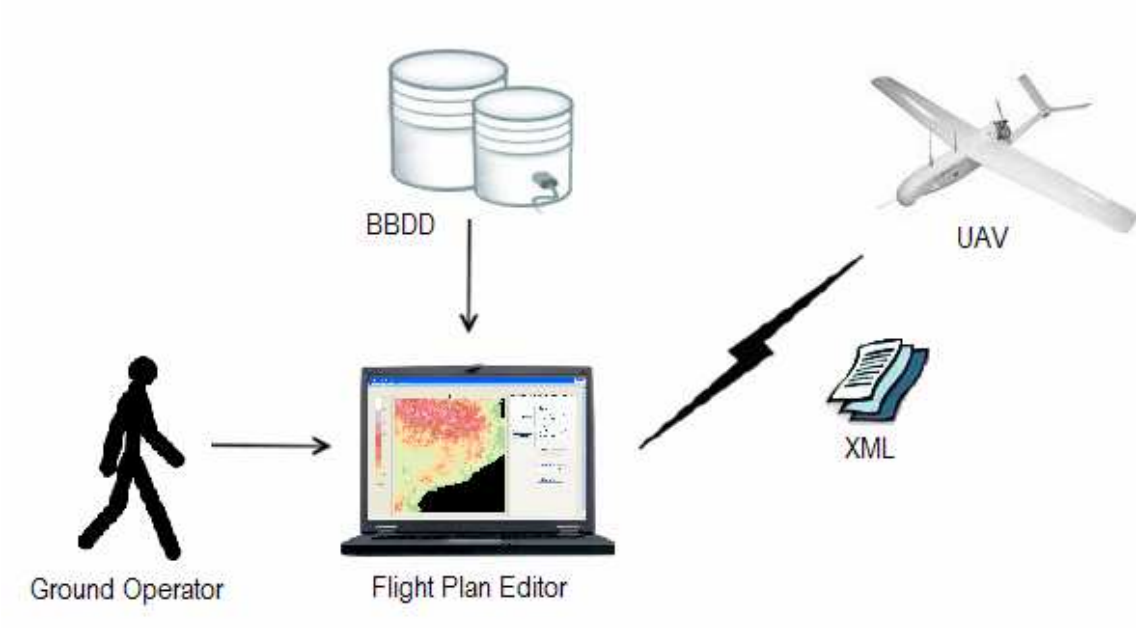


Fig. I.1 Escenario marco de la aplicación

La organización del documento es la siguiente:

Empieza con un capítulo donde se presentan algunos conceptos básicos que se desarrollan a lo largo del trabajo. Conceptos como modelo digital de terreno y algunas definiciones relacionadas con un plan de vuelo son presentados en él.

En el capítulo segundo se presenta la lista de requerimientos del usuario, la arquitectura y el diseño de los distintos módulos que componen la aplicación.

En el capítulo tercero se presenta la implementación de los distintos módulos que componen la aplicación y se exponen detalladamente las clases que contienen. La interfaz de usuario implementada es expuesta también en este capítulo.

El último capítulo del trabajo se dedica a exponer los problemas salvados durante su desarrollo, las líneas futuras con las que se podría mejorar la aplicación y las conclusiones obtenidas después de su realización.

CAPÍTULO 1 . CONCEPTOS PREVIOS

En este capítulo se introducen algunos conceptos necesarios para la comprensión de este documento.

1.1. Modelo digital del terreno

Se denomina modelo digital del terreno (MDT) a una estructura numérica de datos que representa la distribución espacial de una variable cuantitativa, como puede ser la temperatura, la cota, la humedad o la presión atmosférica.

En particular, cuando la variable a representar es la cota o altura del terreno se denomina Modelo Digital de Elevaciones (MDE) o *Digital Elevation Model* (DEM en sus siglas en inglés).

Los modelos digitales del terreno son simbólicos pues establecen relaciones de correspondencia con el objeto real mediante algoritmos o formalismos matemáticos que son tratados mediante programas informáticos.

El modelo digital de elevación es una estructura numérica de datos que representa la distribución espacial de la altitud de la superficie del terreno. Para la obtención de dicho modelo se procesan datos como perfiles, curvas de nivel y cotas altimétricas obtenidos mediante la inspección del terreno. La figura 1.1 ilustra este proceso.

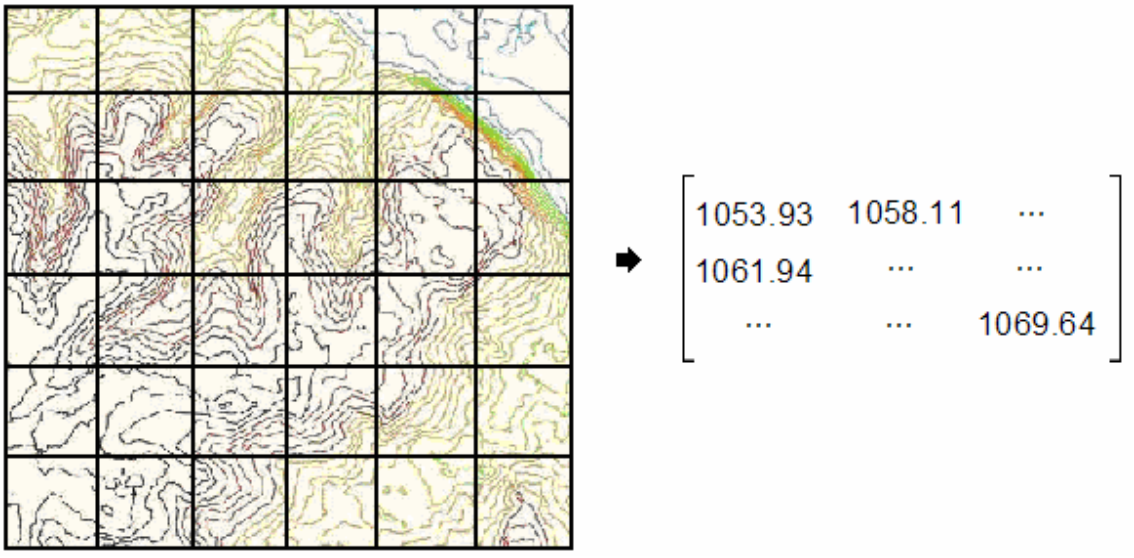


Fig. 1.1 Modelo digital de elevaciones

1.2. Plan de vuelo

En el contexto de este documento el término plan de vuelo se refiere a la trayectoria que debe seguir el UAV y no se refiere al documento que debe ser presentado a las autoridades y que incluye datos como la identificación del avión, horario de salida o otras normas de vuelo que quedan fuera del propósito del presente trabajo.

La figura 1.2 muestra un posible esquema del plan de vuelo que se pretende representar y ejecutar. Este plan de vuelo, que se presenta como ejemplo, está dividido en varias fases o procedimientos. Después de los procedimientos de *Taking Off* (despegue) y *Departure* (salida) el avión se dirige hacia el área de la misión. Una vez allí procede a escanear el área de interés. Cuando ha realizado su cometido en el área se dirige de nuevo al punto de partida. Las últimas etapas del vuelo se corresponden con los procedimientos de A*rrival Route* (ruta de llegada), *Approach* (acercamiento) y *Landing* (aterrizaje).

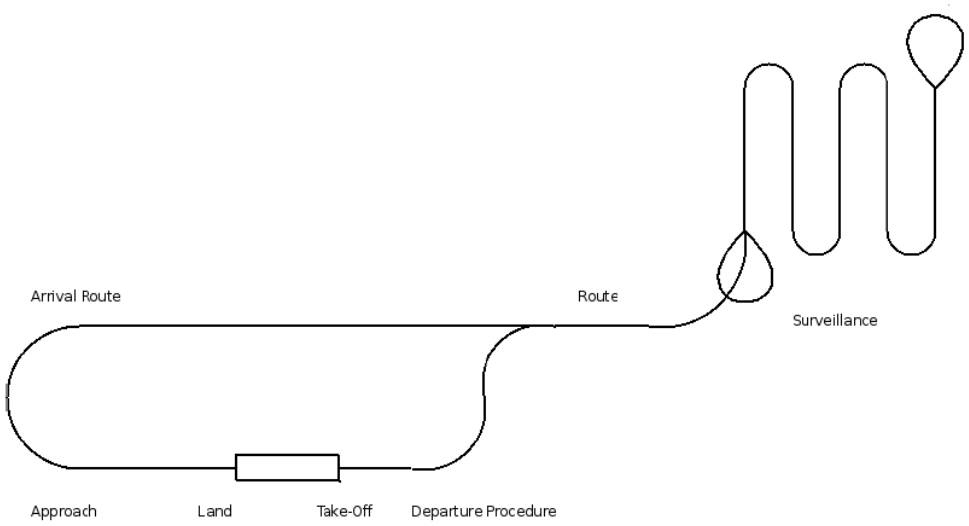


Fig. 1.2 Esquema de un posible plan de vuelo

A lo largo de la redacción de este documento se repiten varios conceptos implicados en la definición y especificación de un plan de vuelo.

Un plan de vuelo constará de varias fases entre las listadas anteriormente. Cada una de estas fases estará compuesta por series de legs.

El concepto de leg ha sido tomado de RNAV [1] y es usado para especificar la trayectoria descrita por el avión para llegar a un punto desde el punto precedente. En el caso más simple la trayectoria será una línea recta pero como veremos es posible definir otras trayectorias.

Por último, un waypoint es, básicamente, una posición geográfica definida en términos de latitud y longitud que debe sobrevolar el UAV.

1.3. Sistemas de Coordenadas

Desde la antigüedad, el hombre ha intentado representar con más o menos precisión la información geográfica que obtenía en sus descubrimientos. A estas representaciones simbólicas se les denomina comúnmente mapas.

Un mapa es una representación gráfica y métrica de una porción de territorio sobre una superficie bidimensional. Que el mapa tenga propiedades métricas significa que ha de ser posible tomar medidas de distancias, ángulos o superficies sobre él y obtener un resultado con una precisión prefijada.

Por otro lado, las coordenadas comúnmente utilizadas en aeronavegación son las coordenadas geográficas. Éstas disponen la situación de un objeto en el espacio tridimensional. Puesto que el editor del plan de vuelo debe trabajar en una pantalla que es bidimensional, para representar un mapa sobre ella se torna imprescindible establecer una correspondencia entre las coordenadas de la pantalla donde se está editando y las coordenadas geográficas sobre la superficie de la tierra.

En los siguientes apartados se presentan los distintos sistemas de coordenadas utilizados.

1.3.1. Coordenadas geográficas

Este sistema de referencia, conocido de antiguo, es el empleado en navegación marítima y aérea para localizar un punto sobre la superficie de la Tierra. Su principal virtud es que abarca toda la superficie del globo terráqueo, lo que simplifica la representación de localidades muy alejadas entre sí. Pero sobre papel no resulta demasiado práctico debido a la curvatura sobre el mapa de las líneas de referencia.

La Tierra tiene una forma aproximada de elipsoide de revolución (alrededor del eje norte-sur) con semieje menor de 6357 Km. y semieje mayor (en la circunferencia del Ecuador) de 6378 km. La posición de un punto sobre su superficie viene dada con referencia a una malla de círculos paralelos y meridianos.

1.3.1.1. Latitud

Se denomina latitud a la distancia angular, medida sobre un meridiano, entre la línea ecuatorial y el paralelo de una localización terrestre. Se mide en grados. Si el punto pertenece al hemisferio norte es positiva y negativa para el hemisferio sur. Variará entre 0º y 90º norte y entre 0º y - 90º sur.

También se puede describir como la distancia angular entre cualquier punto de la Tierra y el paralelo 0 o Ecuador. Puede ser norte o sur. Se mide en grados de 0 a 90 a partir del Ecuador, tanto al norte como hacia el sur.

1.3.1.2. Longitud

La longitud expresa la distancia angular, medida paralelamente al plano del Ecuador terrestre, entre el meridiano de Greenwich y un determinado punto de la Tierra. Existen varias maneras de expresar la longitud pero en este trabajo se utiliza la siguiente nomenclatura: entre -180 y 180 º, siendo positiva hacia el este o negativa hacia el oeste.

1.3.1.3. Altitud

Para especificar una posición completamente, debajo o sobre la superficie de la Tierra, es necesario especificar también la elevación o altitud. Esta puede ser expresada relativa a un dato como el nivel del mar o la distancia al centro de la Tierra. Lo más común es hacerlo respecto al nivel medio del mar.

La figura 1.3 muestra, a modo de ejemplo, la posición del punto P con coordenadas 71º 3’ 27” E de longitud (λ) y 42º 21’ 30” N de latitud (W) así como los orígenes de coordenadas del sistema.

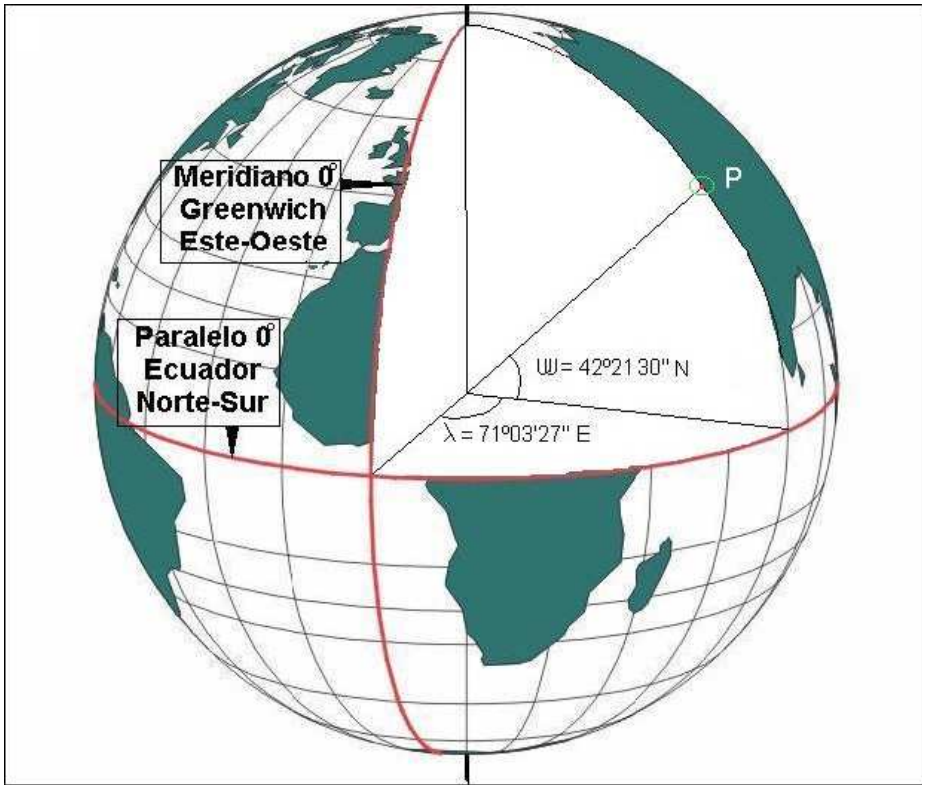


Fig. 1.3 Coordenadas geográficas

1.3.2. Coordenadas UTM

El sistema de coordenadas Universal Transversa de Mercator (En inglés *Universal Transverse Mercator*, UTM) es un sistema de coordenadas basado en la proyección geográfica transversal de Mercator.

Dado que las coordenadas geográficas determinan puntos sobre un elipsoide de revolución definido, para pasar a un plano habrá que establecer la adecuada correspondencia entre los puntos del elipsoide y el plano. Esta transformación de un espacio tridimensional a uno bidimensional se denomina proyección. Uno de los sistemas de proyección más empleados es el sistema de proyección UTM.

Se considera que:

- La Tierra queda dividida en 60 husos. Podemos hablar del huso 30, del huso 31, etc. Así se limita la proyección a un huso de 6 grados de longitud reduciendo la deformación lineal.
- Los husos se numeran correlativamente del 1 al 60 a partir del "antimeridiano de Greenwich (180 º) y en sentido creciente hacia el Este.
- Cada huso se divide horizontalmente, entre 84º de latitud Norte y los 80 º de la latitud Sur, en 20 bandas entre paralelos. Considerando que las latitudes van de 80º (84) de latitud Norte y los 80 º de la latitud Sur, tenemos 160/8 = 20 bandas denominadas de sur a Norte con las letras C a la X (se excluyen las letras I, Ñ y O) (la C sería la 80ºS a 72ºS) y la X (sería de 72ºN a 84ºN). Las bandas C a M están en el hemisferio sur y las bandas N a X están en el hemisferio norte. Nos sirve de norma nemotécnica recordar que cualquier banda que esté detrás de la N en el alfabeto N (de norte) está en el hemisferio norte. Las primeras 19 bandas (C a W) están separadas o tienen una altura de 8° cada una. La banda 20 o X tiene una altura de 12°.
- Cada huso queda así delimitado en áreas de 6º de longitud y 8º de latitud que se denominan zonas y constituyen la cuadrícula básica de la proyección UTM.

Para realizar la proyección UTM se supone un cilindro cuyo diámetro coincide con el diámetro del ecuador y es tangente al globo terrestre en el meridiano central al huso a representar. La figura 1.4 muestra esta situación.

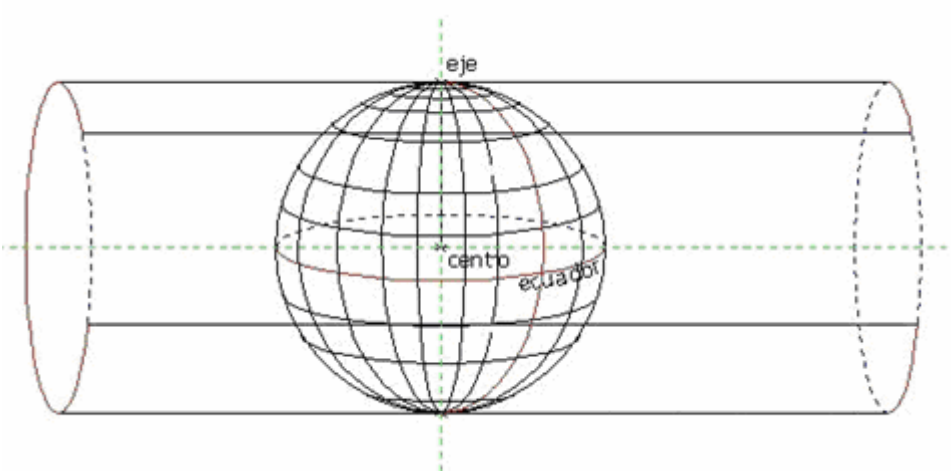


Fig. 1.4 Cilindro empleado en la proyección UTM

La figura 1.5 muestra la zona T del huso 31 en el centro que es la zona a la que pertenece toda la extensión del territorio de Catalunya.

Fig. 1.5 Zona 31T Catalunya

Dado que la forma de la tierra no es regular, es un elipsoide algo achatado en los polos, la proyección de su superficie sobre un mapa deberá tener en cuenta las irregularidades que presenta. Estas varían dependiendo de cada región.

Para ello se definen para cada región de la Tierra los siguientes conceptos:

- Elipsoide : que no es más que el resultado de revolucionar una elipse sobre su eje. Queda definido por la longitud de sus dos ejes. Cada región emplea un modelo matemático distinto con el fin de adaptarse mejor a la forma de la tierra en la zona a cartografiar.

- Geoide: Se define como la superficie teórica de la tierra que une todos los puntos que tienen igual gravedad. La forma así creada supone la continuación por debajo de la superficie de los continentes, de la

superficie de los océanos y mares suponiendo la ausencia de mareas, con la superficie de los océanos en calma y sin ninguna perturbación exterior. Se consideran perturbaciones exteriores la atracción de la luna (mareas) y las interacciones con todo el sistema solar.

Finalmente, se define como punto fundamental de un datum como el punto tangente al elipsoide y al geoide, donde ambos son coincidentes. La figura 1.6 ilustra este concepto.

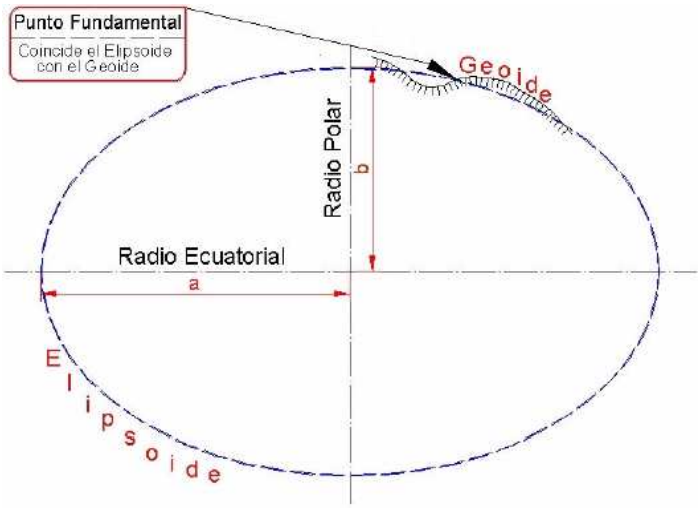


Fig. 1.6 Punto fundamental

Cada datum está compuesto por:

- Un elipsoide, definido por sus dos ejes.
- El punto llamado fundamental en el que el elipsoide y el geoide son tangentes. Este punto fundamental se define por sus coordenadas geográficas de longitud y latitud, además del acimut de una dirección con origen en el punto fundamental. Esta desviación queda definida por dos parámetros Eta (desviación en la vertical) y Xi (desviación en el meridiano).

Definido el datum ya se puede elaborar la cartografía de cada lugar, pues se tienen unos parámetros de referencia que relacionan el punto origen del geoide y del elipsoide con su localización geográfica, así como la dirección del sistema.

Para Catalunya y España la mayor parte de la cartografía perteneciente al Institut Cartogràfic de Catalunya y al Instituto Geográfico Nacional se encuentra referenciada con el European Datum – 1950, más conocido por sus siglas ED50. Bajo este datum se localiza la península Ibérica, el archipiélago Balear así como Ceuta y Melilla. Este datum toma como referencia el “Elipsoide Internacional” también llamado “Elipsoide de Hayford” con base en la Torre de Helmert (Postdam, Alemania).

Designación coordenadas UTM

Finalmente, las coordenadas UTM consisten en un par de coordenadas denominadas easting (x) y northing (y) que indican la posición sobre la rejilla formada por la proyección UTM.

Únicamente con las coordenadas del punto su situación no queda definida. Los siguientes datos deben ser indicados junto a la designación de las coordenadas:

- Sus unidades.
- El hemisferio donde se encuentran.
- El huso UTM de proyección.
- El datum utilizado en la proyección.

El origen de coordenadas del sistema es distinto para cada huso, tomándose como origen la intersección del meridiano central del huso con el Ecuador. En el hemisferio norte, toma un valor en *x* de 500.000 metros y un valor en *y de* 0 metros. De esta manera se evita que el sistema genere coordenadas negativas en el hemisferio norte. Los valores crecen en x hacia el este y crecen en y hacia el Norte.

El formato de designación de la cuadrícula depende de la resolución con que se encuentran las coordenadas UTM. La figura 1.6 muestra una coordenada UTM designada con una resolución de 1 metro:

Fig. 1.7 Designación UTM para resolución 1m

Hacer notar también que, tal y como están tomadas las referencias, las coordenadas northing van a ser por lo general más grandes ya que distarán más del origen (recordemos que es el Ecuador). La máxima distancia será la longitud de un arco de 90º con el radio de la Tierra. En resumen, nunca van a superar los 10.000 Km. (7 cifras si medimos en metros). Las coordenadas easting , por otra parte, no van a superar nunca (siempre que nos movamos dentro de la zona que usamos como referencia) los 1000 Km. En realidad la longitud horizontal máxima de una zona, que se da en el Ecuador, es de unos 670 Km., por lo que nunca utilizaremos más de 6 cifras si medimos en metros.

CAPÍTULO 2. DISEÑO Y ARQUITECTURA

En este capítulo se presenta la arquitectura de la aplicación y se expone el diseño de los módulos que la componen así como sus funcionalidades.

Antes de presentar los módulos diseñados se incluye un apartado de requerimientos de la aplicación en el cual se presentan conceptos a considerar durante el diseño y la implementación del mismo.

2.1. REQUERIMIENTOS

Se ha elaborado una lista de requerimientos para la presente aplicación.

- La aplicación debe desarrollarse íntegramente en entorno .NET y lenguaje de programación C# y tener un aspecto de aplicación Windows estándar.

En líneas generales el módulo DEM:

- Debe ser capaz de representar un área geográfica a partir de la información de su altura suministrada por el Institut Cartogràfic de Catalunya.

- Debe ser capaz de realizar zoom y pan sobre esta área. Se deben depurar estos procesos para que el tiempo de espera resulte lo más reducido posible.

- El tamaño del área de visualización será de 650x650 píxeles.

- La aplicación se debe inicializar con una visión general de todo el territorio de Catalunya incluido en el modelo digital de elevaciones.

- Se debe dotar al mapa de una leyenda y una escala. Los colores utilizados para la representación serán acordados con el usuario final.

El módulo encargado de la gestión del plan de vuelo:

- El modelo del plan de vuelo implementado debe seguir los conceptos definidos por el grupo ICARUS basados en RNAV o navegación de área extraídos de [1].

- Ser capaz de introducir un plan de vuelo directamente sobre la representación gráfica.

- Generar un documento con toda la información relevante del plan de vuelo introducido. El formato del fichero de salida será XML.

- Debe ser capaz de representar un plan de vuelo introducido en formato XML.

- El editor debe permitir editar y modificar las relaciones existentes entre cada entidad que compone el plan de vuelo. Se vuelve indispensable implementar un editor donde todas estas relaciones puedan ser establecidas y modificadas.

- Además se debe poder verificar que las relaciones entre las distintas entidades que componen el plan de vuelo estén correctamente establecidas.

- Las relaciones establecidas deben poder adaptarse a la modalidad de vuelo reverso. Es decir, una vez establecida la trayectoria para volar de un origen a un destino la aplicación debe ser capaz de verificar el trayecto del destino al origen sin necesidad de editar el nuevo trayecto.

2.2. ARQUITECTURA

La arquitectura de la aplicación se ha basado en un modelo de dos capas separando así las funcionalidades en distintos bloques que se comunican entre sí. La figura 2.1 ofrece una visión general de la arquitectura implementada para la aplicación.

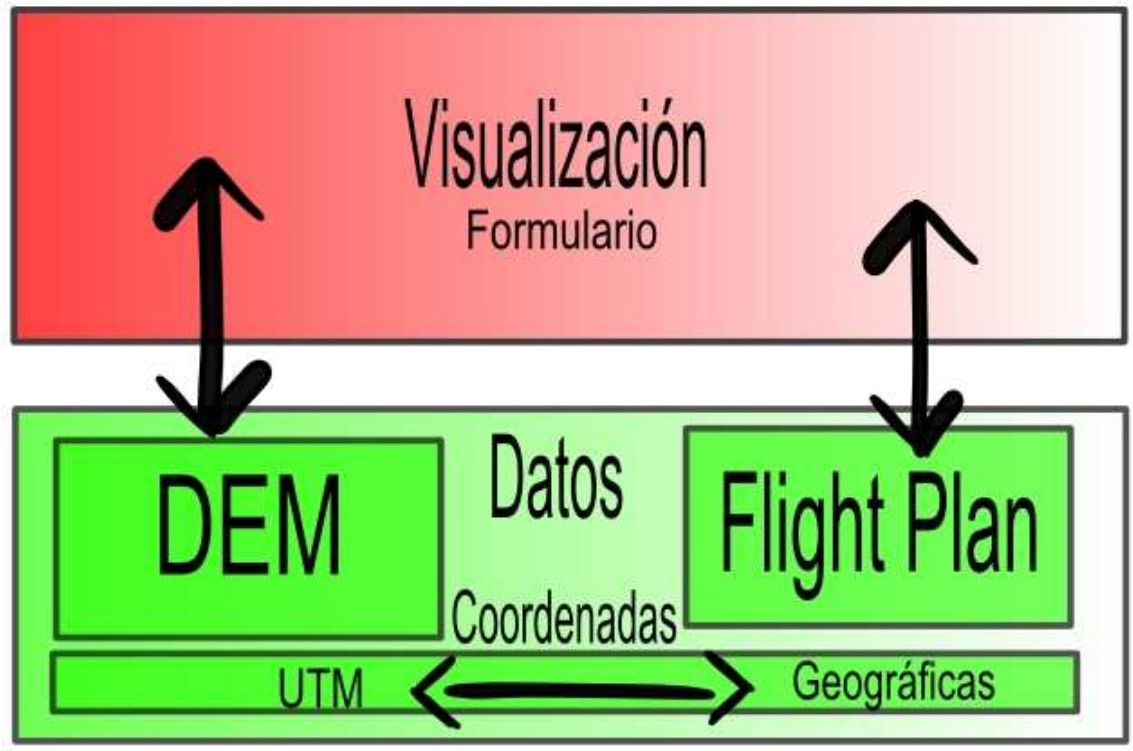


Fig. 2.1. Arquitectura de la aplicación

Sigue a continuación una breve descripción de las funcionalidades de cada capa:

Capa de Visualización

La capa de visualización está formada por la interfaz de usuario. Básicamente consiste en un formulario y el resto de controles asociados a él que permiten al usuario introducir datos y comandos a la aplicación. La capa de visualización:

- Muestra los datos requeridos por el usuario, ofrece información contextual y modifica el estado de la interfaz según la acción que emprenda el usuario.
- Visualiza los datos del DEM según la vista establecida por el usuario.
- Muestra el plan de vuelo.

- Contiene la lógica para el formato y validación de las entradas del usuario. Para ello contiene métodos encargados de procesar los datos introducidos por el usuario y establecer relaciones entre las distintas entidades que componen el plan de vuelo. En el apartado 2.3.2 se exponen con detalle todas estas relaciones.

Capa de Datos

Esta capa contiene dos módulos bien diferenciados:

- El módulo DEM recoge los datos del modelo digital del terreno y las clases encargadas de procesarlo. Contiene los métodos de acceso al fichero del modelo digital del terreno. Se encarga de acceder a ellos y volcarlos a una clase intermedia antes de proceder a su visualización.

- El módulo denominado Flight Plan almacena el plan de vuelo y gestiona las clases asociadas al mismo.

Subyacen además bajo todas estas capas las clases y métodos encargados de realizar las proyecciones entre los distintos sistemas de coordenadas utilizados.

2.3. DISEÑO

Se desarrolla en el siguiente apartado el diseño de los distintos módulos que componen la aplicación.

2.3.1. Módulo DEM

Se ha dotado a la aplicación de un visualizador del modelo digital del terreno de Catalunya capaz de hacer zoom y pan sobre un área geográfica determinada. El propósito del módulo es permitir la visualización de mapas de elevación de terreno que ayuden al operador de tierra a diseñar el plan de vuelo.

Para llevar a cabo este propósito se ha partido de un modelo digital de terreno suministrado por el ICC (Institut Cartogràfic de Catalunya).

Los datos incluidos en el modelo están referidos al sistema geodésico de referencia oficial (European Datum 1950) y en proyección Universal Transversa Mercator (Huso 31).

Consta de dos ficheros. Un fichero recoge las cotas altimétricas de toda Catalunya equiespaciadas 30 metros para la mitad este y el segundo fichero para la mitad oeste. También se incluyen datos del sur de Francia y el este de Aragón con el mismo equiespaciado.

El formato de los ficheros .txt es el formato grid ASCII de Arc/Info y constituye un modelo de elevación del terreno de malla regular en formato ASCII.
Cada uno de ellos tiene una cabecera donde se indican el número de filas (8902) , número de columnas (4618), las coordenadas del vértice sudoeste, las coordenadas del vértice noreste, el paso de malla (30m) y el símbolo utilizado para las superficies de las que no se tienen datos precisos (-9999.99).

A continuación vienen los datos de las cotas por filas de norte a sur y cada fila de oeste a este. Las cotas vienen dadas en metros con dos decimales y separadas por blancos. A modo de ejemplo se hacen constar los primeros valores que aparecen en uno de los ficheros : 1053.93 1058.11 1061.94 1065.82 1069.64 ... Cada fichero consta de 41.109.436 cotas repartidas en 8902 filas y 4618 columnas.

La zona que cubre cada fichero es:

Nombre fichero	CATA30_OEST_REV1 .TXT
VERTEX SW :	257995.00, 4484975.00
VERTEX NE :	396505.00, 4752005.00

Nombre Fichero	CATA30_EST _REV1.TXT
VERTEX SW :	396505.00, 4484975.00
VERTEX NE :	535015.00, 4752005.00

Las zonas que cubre el modelo digital del terreno son las marcadas en azul y verde en la figura 2.2.

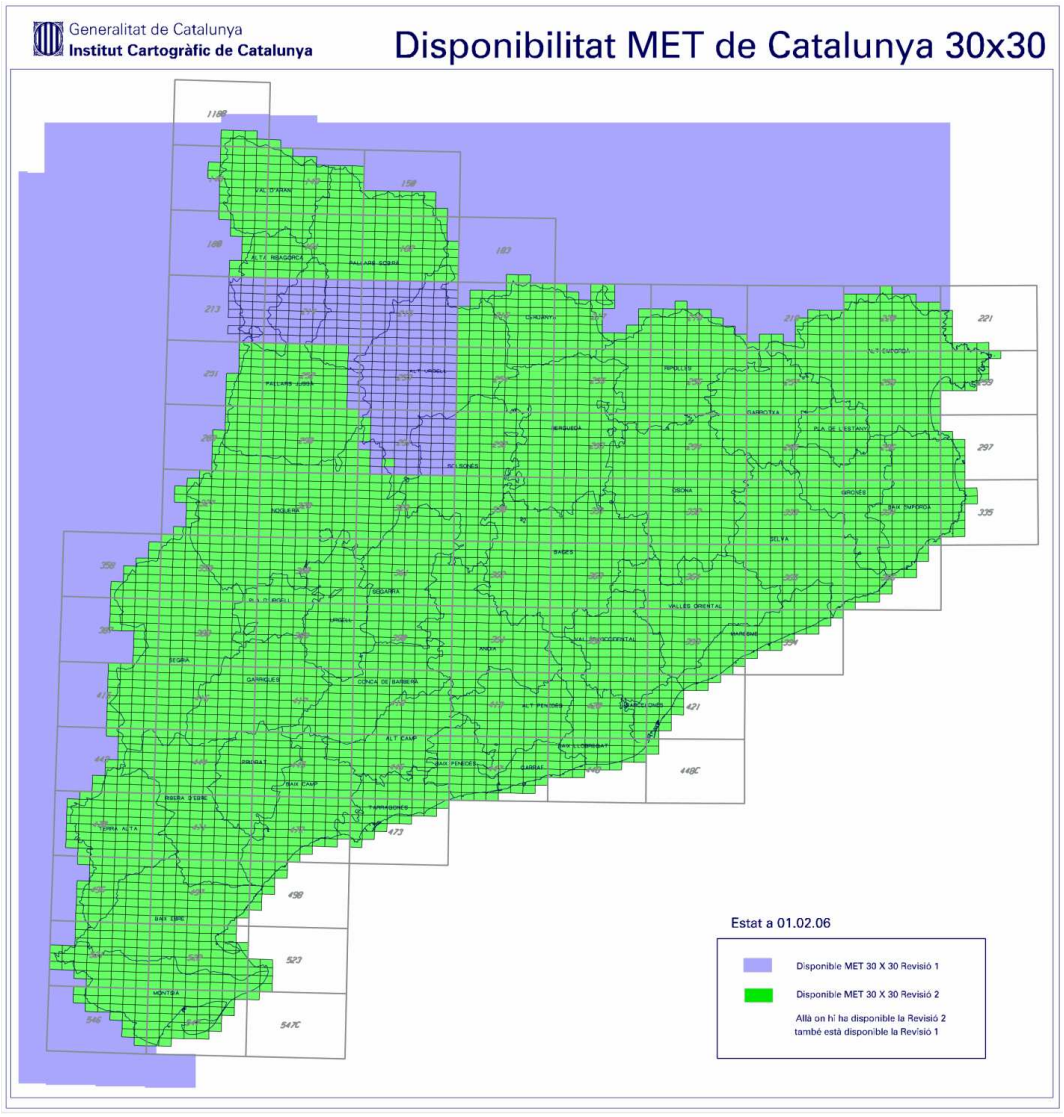


Figura 2.2 Áreas cubiertas por el modelo digital de terreno

2.3.2. Diseño del modelo del plan de vuelo

Uno de los requisitos del usuario consiste en que el modelo de plan de vuelo se corresponda con los conceptos basados en el sistema de navegación de área (RNAV) utilizados por el grupo Icarus.

La navegación aérea convencional en Europa continental se apoya en el uso de las redes terrestres de radioayudas que definen, a su vez, la estructura de una rutas que transitan las aeronaves (rutas ATS). Este sistema de rutas es, por tanto, fijo e inflexible desde un punto de vista geográfico. El sistema de rutas no permite explotar las tecnologías de los nuevos equipos de navegación aérea.

La navegación de área (RNAV) permite la planificación y diseño de rutas no apoyadas en ayudas a la navegación convencionales, proporcionando una mayor capacidad y flexibilidad en la utilización del espacio aéreo disponible. Entre estos aspectos cabe destacar la posibilidad de definir trayectorias más directas y el consiguiente ahorro de combustible.

Durante la fase de diseño del presente módulo es necesario identificar los diferentes elementos que componen el plan de vuelo y definir las relaciones existentes entre ellos. Las descripciones y relaciones entre los conceptos expuestos a continuación deberán ser editables, modificables y verificables utilizando el presente editor.

2.3.2.1. Plan de vuelo

El plan de vuelo especifica el camino a seguir por el avión. Cada plan de vuelo queda especificado por una serie de fases (stages) que se deberán suceder en orden.

El plan de vuelo principal es aquel que incluye la ruta a seguir por el avión desde principio a fin. Otros planes de vuelo adicionales pueden ser incluidos como planes parciales, donde no todas las fases estarán incluidas. El propósito de los mismos es dotar al avión de rutas alternativas en caso de emergencia.

2.3.2.2. Fases

Las fases organizan los legs en secuencias y constituyen una agrupación de legs de alto nivel. Es por eso que cada fase del plan de vuelo debe corresponder con una fase bien definida del mismo y los legs incluidos en ella deben perseguir un propósito común. Los diferentes tipos de fases que puede tener un plan de vuelo se detallan a continuación:

- Taxi: Desplazamiento hasta la pista.
- TakeOff: Legs usados durante el despegue.
- Departure Procedure: Comprende los legs del procedimiento de salida.
- Route: Legs para navegar desde un punto inicial hasta un punto destino.
- Surveillance: Legs a seguir durante una operación de vigilancia.
- ArrivalRoute: Contiene la ruta a seguir antes de iniciar la aproximación.
- Approach: Procedimineto de aproximación a la pista.
- Land: Operación de aterrizaje.

Se ha comentado anteriormente que las fases de un determinado plan de vuelo deben ocurrir en secuencia. Por ello cada fase, excepto la primera y la última, tienen una fase precedente y otra fase que le sigue.

Además cada fase puede tener más de una salida y más de una entrada. Por eso el número de salidas de una fase debe coincidir con el número de entradas de la fase que le sigue y sus entradas con las salidas de la fase precedente. Por esta razón los legs iniciales y finales de cada fase deberán ser reconocibles visualmente mientras se está editando.

2.3.2.3. Legs

Un leg especifica la trayectoria seguida por el avión entre dos waypoints.

En el caso más simple la trayectoria será una línea recta pero para definir un plan de vuelo real es necesario poder definir de antemano otros tipos de trayectorias que permitan definir y ejecutar trayectorias más complejas.

Cada leg creado deberá tener una referencia al siguiente leg que seguirá el UAV y una referencia al leg anterior. Aunque la referencia al leg anterior pueda parecer superflua ésta es necesaria para poder implementar la modalidad de vuelo reverso.

Un leg inicial será aquel que no tenga ningún leg previo. De la misma manera un leg final será aquel que no tenga ningún leg siguiente.

Legs Básicos

A continuación se exponen las trayectorias básicas:

- Track to fix: se corresponde con una trayectoria en línea recta entre un waypoint y otro. La figura 2.3 muestra el símbolo usado en las cartas de navegación RNAV para mostrar un leg del tipo Track to fix entre los waypoints A (origen) y B (destino).

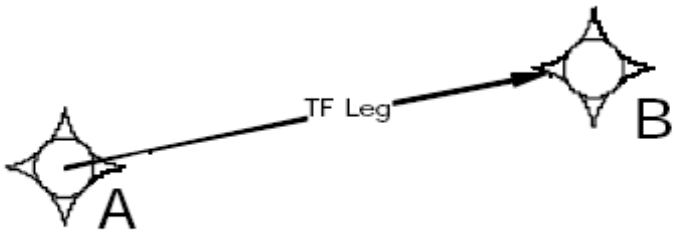


Fig. 2.3 Leg Track to fix

- Direct to fix: se corresponde con una trayectoria en línea recta desde cualquier área inicial. Es decir, ordena al UAV dirigirse directamente al waypoint destino sea cual sea la posición actual del avión en el momento de recibir la orden.

- Radius to fix: Definido como una trayectoria circular alrededor del centro de giro que termina en el waypoint destino del leg. La figura 2.4 muestra el símbolo RNAV para este tipo de leg.

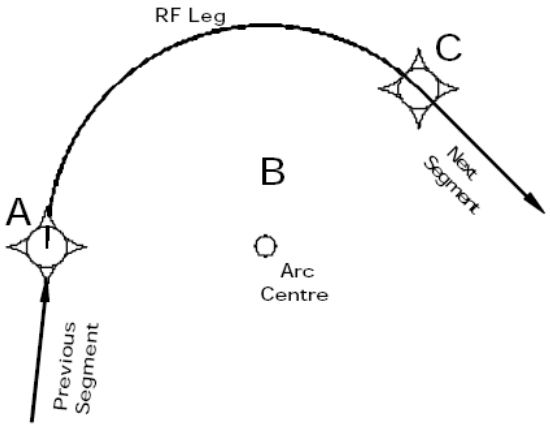


Fig. 2.4 Leg Radius to fix

- Hold Pattern: Define una trayectoria elíptica sobrevolando el waypoint destino del leg. La figura 2.5 muestra el símbolo RNAV para este tipo de leg.

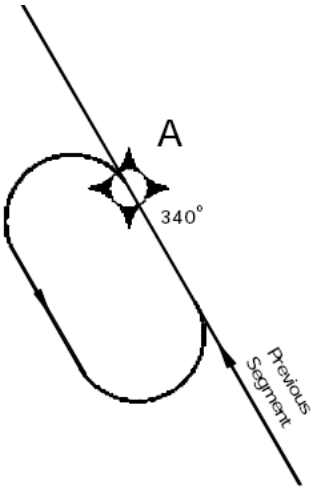


Fig. 2.5 Leg Hold Pattern

Intersection Legs

Un leg intersección indica puntos donde dos o más caminos coinciden. Es, en definitiva, un punto donde ciertas condiciones deben ser evaluadas para decidir cual va a ser la siguiente trayectoria que tomará el avión.

Cada leg definido como intersección debe tener asociada una condición que será evaluada cuando el avión llegue hasta el punto de intersección para decidir que camino de entre todos los posibles tomar. De no cumplirse ninguna de las condiciones, el UAV seguirá un leg asignado por defecto.

En resumen todos los legs básicos tienen solo un leg que les precede y otro que les sigue mientras que en el caso de una intersección existe más de un posible leg siguiente.

Es necesario que todos los legs que converjan en un punto sean incluidos en un leg intersección. Usar un leg Intersección cuando dos caminos convergen puede parecer innecesario pero el requerimiento del usuario de volar los legs en el sentido opuesto a como se han definido hace aconsejable considerar este tipo de intersecciones. Será necesario definir de manera análoga a los legs siguientes todos los legs previos a la intersección para poder permitir el vuelo en el sentido opuesto al definido.

2.3.2.4. Waypoint

Cada waypoint de una ruta RNAV representa una localización donde el sistema de navegación del avión debe efectuar una transición automática entre un leg y otro.

Un waypoint no sólo recoge las coordenadas geográficas del punto a sobrevolar sino que además incluye datos como el tipo de transición que debe ejecutar el piloto automático al aproximarse al punto y datos de altitud y velocidad a las que se debe sobrevolar. Esta transición puede incluir cambios de la velocidad y/o altura y/o dirección.

En cada waypoint se debe especificar qué tipo de transición se realizará. Existen dos tipos de transiciones básicas:

- Transiciones fly-by: aquellas en las que el sistema de navegación anticipa el giro hacia la trayectoria que especifica el siguiente leg.

- Transiciones fly-over: en las que el sistema de navegación sobrevuela el waypoint especificado antes de iniciar la maniobra hacia el siguiente leg.

2.3.3. Diseño de la interfaz

Se recogen en este apartado los criterios de diseño que se han aplicado para el diseño de la interfaz de usuario.

- El tamaño del área de visualización deberá de 650x650 píxeles. El requerimiento del usuario sobre el tamaño del área dedicada a la visualización se debe a que es necesario que el área cuando el máximo factor de zoom de la aplicación se esté utilizando sea de unos 20 Km cuadrados. El gran tamaño dedicado a la imagen hace que las funciones de zoom y pan deban ser depuradas para que el tiempo de espera al hacer uso de ellas no sea excesivamente elevado.

- La aplicación constará de un único formulario para evitar así abrir formularios secundarios que entorpezcan la visión del usuario del plan de vuelo que está editando.

- El espacio dedicado a los controles de edición se situará en la parte derecha de la pantalla. Recogerá todas las funcionalidades presentes en el editor e irá cambiando su aspecto según la tarea que desee desempeñar el usuario.

- Estas tareas estarán estrechamente ligadas con las distintas entidades que componen el plan de vuelo. Así , se intentará recoger todas las funcionalidades relacionadas con una entidad del plan de vuelo (waypoints, legs, fases, etc ...) , siempre que el espacio disponible en pantalla lo permita, en una misma agrupación. Para ello se situa en la parte derecha de la pantalla una agrupación de pestañas que permite la selección de la tarea que se quiere realizar. A modo de ejemplo y aunque estas funcionalidades se detallan en el cápitulo tercero, la pestaña leg agrupa los controles necesarios para introducir, seleccionar y editar un leg mientras que la pestaña map recogerá los controles necesarios para ajustar las herramientas de zoom y pan de las que dispone la aplicación. De esta manera el uso de las distintas funcionalidades se hace algo más intuitivo.

- Como es habitual en las aplicaciones Windows la barra de herramientas se situará en la parte superior izquierda de la pantalla y dará acceso a algunas de las funcionalidades más utilizadas durante la edición del plan de vuelo.

- La leyenda utilizada será siempre visibles y se mostrará a la izquierda del mapa.

La figura 2.6 muestra la disposición de los componentes sobre la pantalla:

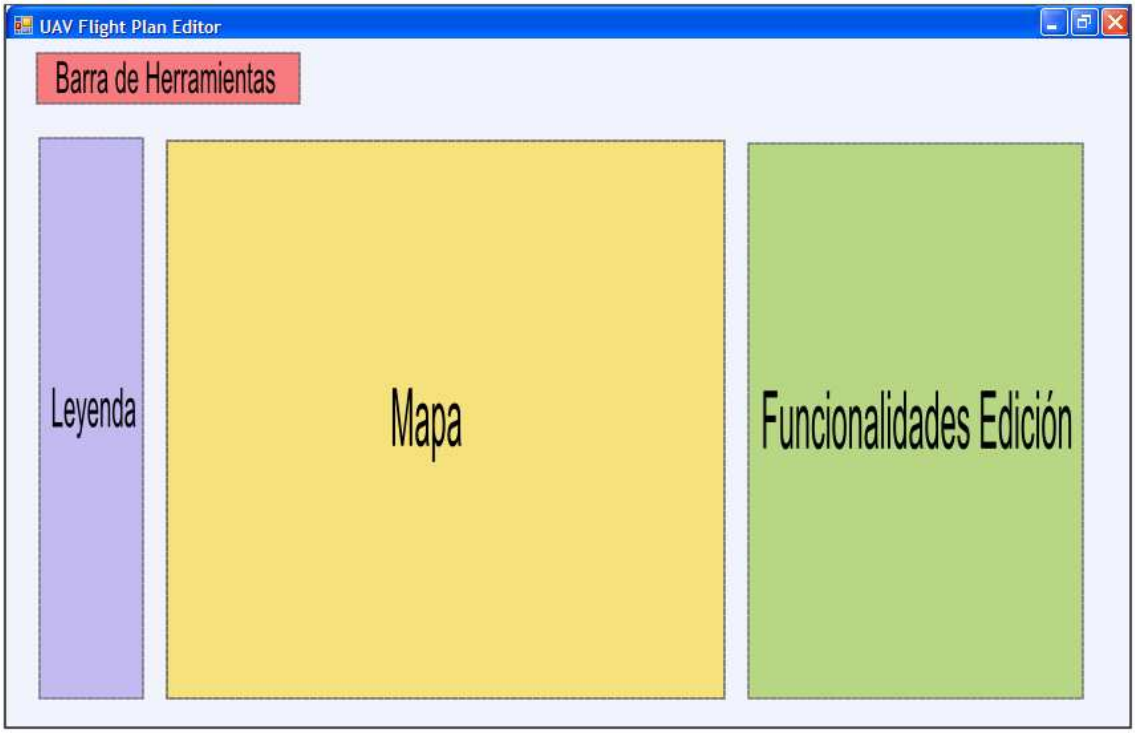


Fig. 2.6 Diseño Interfaz

CAPÍTULO 3 . IMPLEMENTACIÓN

En este capítulo se presentan la implementación de las capas que componen la aplicación y de los distintos módulos que contienen. Se exponen en detalle las distintas clases que los componen. La figura 3.1 muestra el diagrama de clases resultante.

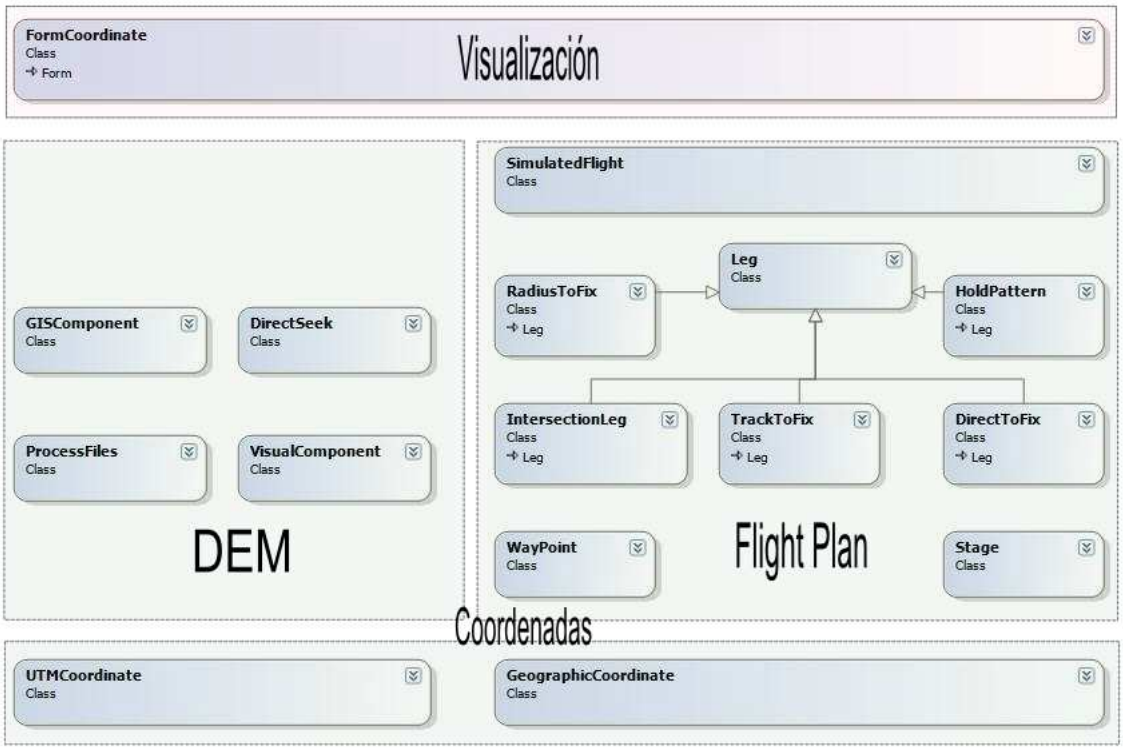


Fig. 3.1 Diagrama de clases de la aplicación

3.1. Implementación del módulo DEM

Para la realización del módulo DEM se ha tenido en cuenta la separación de funcionalidades del mismo. Una clase se encarga de la lectura de los datos y otra es la encargada de su procesado gráfico permitiendo así la mejora de cualquiera de las partes sin afectar a la otra.

Antes de emprender la implementación de estas clases se modificaron los datos suministrados por el ICC tal y como se resume en el siguiente apartado.

3.1.1. Preprocesado de los datos del modelo digital del terreno

Dada la fragmentación de los datos suministrados inicialmente por el ICC y por requerimientos de eficiencia de la aplicación, para agilizar el proceso de lectura de las cotas altimétricas, se preprocesaron ambos ficheros unificándolos en uno solo donde cada cota ocupa una longitud determinada en bytes. Cada cota pasa así a ocupar dos bytes y de esta manera el acceso al fichero es más

rápido ya que dadas las coordenadas de un punto es posible calcular la posición en el fichero de la cota buscada y realizar la lectura de los datos de interés con mayor agilidad. Además procesando los datos originales se ha conseguido reducir su tamaño de 600 MB iniciales a los 300MB.

La clase encargada de realizar la lectura inicial de los dos ficheros es la clase GISComponent. Sus atributos son mostrados en la figura 3.2. Esta clase lee los datos contenidos en los dos ficheros y los almacena temporalmente en la matriz MyAltitudeMatrix(). Posteriormente los datos son volcados en un fichero único suministrado junto a la aplicación con nombre Catalonia.dat.

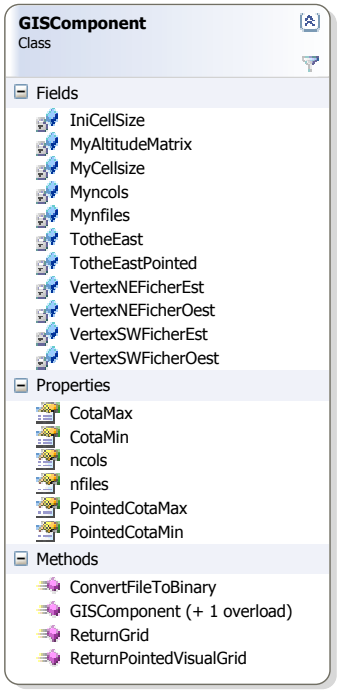


Fig. 3.2 Clase GISComponent

Aunque la creación del fichero sólo deba ser ejecutada una vez se ha conservado el código encargado del procesado en el constructor de la clase ProcessFiles. La figura 3.3 muestra su diagrama.

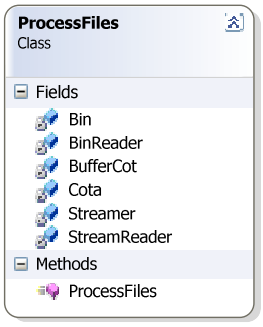


Fig. 3.3 Clase ProcessFiles

3.1.2. Clases principales del módulo DEM

3.1.2.1. Clase DirectSeek. Lectura del fichero

La clase DirectSeek es la encargada de la lectura del fichero cada vez que el usuario cambia la vista mediante el uso de las herramientas zoom o pan de las que dispone la aplicación.

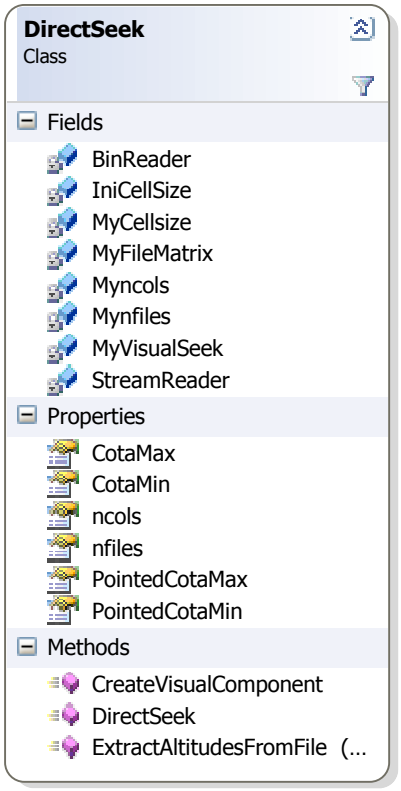


Fig. 3.4 Clase DirectSeek

Tabla 3.1. Atributos y métodos de la Clase DirectSeek

IniCellSize	Variable entera que indica la resolución de la rejilla UTM. Toma el valor de 30 metros.
MyCellSize	Variable entera que indica el tamaño en metros que representa cada píxel en la vista actual.
MyFileMatrix	Matriz que contiene los valores de las cotas altimétricas extraídas del fichero
BinReader	Variable utilizada para leer las cotas almacenadas en formato binario.
ExtractAltitudesFromFile()	Método encargado de la lectura del fichero

Todo el código encargado de la lectura del fichero está contenido en el método ExtractAltitudesFromFile() . A este método se le pasan como parámetros el vértice superior izquierdo y el factor de zoom de la vista actual. La aplicación

realiza la lectura del fichero solicitada. El resultado de la llamada a este método es un objeto de la clase VisualComponent debidamente cumplimentado con los datos requeridos por el usuario.

3.1.2.2. Clase VisualComponent

Es la clase encargada de almacenar los datos extraídos del fichero temporalmente y componer el bitmap que se mostrará por pantalla de acuerdo con la leyenda utilizada. Cada representación gráfica de los datos obtenidos del DEM se corresponde con un objeto de esta clase.

Los datos de las altitudes son procesados para convertirlos en un bitmap que posteriormente se vuelca en la imagen central.

En una primera realización se optó por utilizar métodos de la librería gráfica GDI que ofrece .NET para crear y modificar imágenes bitmap. El código de esta implementación se recoge en el método ConvertToColorBitmap().

Comprobando que se trataba de un proceso muy lento, se optó por descartar el código y realizar un segundo procesado de las cotas altimétricas. Aunque el uso de punteros en C# no es habitual en este caso se optó por acceder directamente a las posiciones de memoria que ocupa la imagen utilizando un método que se conoce como código *unsafe*. Trabajando con código *unsafe* se obtiene una mayor velocidad de procesado y se consumen menos recursos que trabajando con GDI. El código de esta segunda realización se recoge en el método ProcessBitmap().

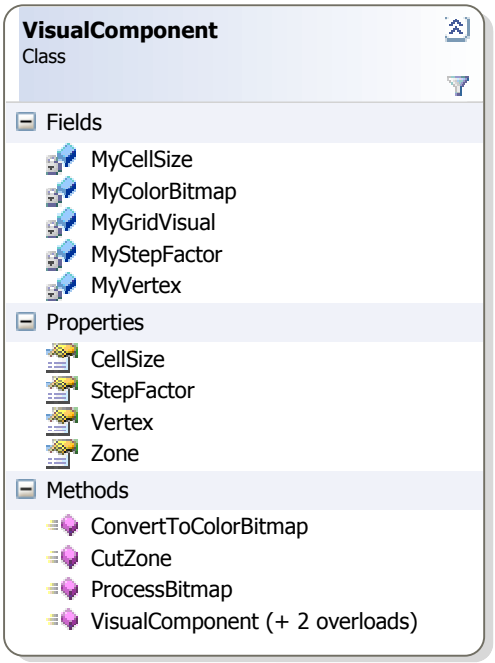


Fig. 3.5 Clase VisualComponent

Tabla 3.2. Atributos y métodos de la Clase VisualComponent

MyCellSize	Variable entera que indica el tamaño en metros que representa cada píxel en la vista actual.
MyColorBitmap	Bitmap. En él se compone la imagen a mostrar.
MyGridVisual	Matriz que contiene las cotas altimétricas.
MyStepFactor	Variable entera que indica cada cuantas cotas altimetricas se toma una muestra de la orografía del terreno.
MyVertex	Variable del tipo UTMCoordinate que almacena las coordenadas del vértice superior izquierdo de la imagen.
ConvertToColorBitmap()	Método que utiliza métodos de la librería GDI de .NET para transformar la matriz de cotas en la imagen del mapa.
ProcessBitmap()	Método que utiliza código unsafe para transformar la matriz de cotas en la imagen bitmap.
CutZone()	Función implementada para permitir la transferencia de la matriz de enteros leída del fichero a futuras aplicaciones.

3.1.3. Funcionalidades del módulo DEM

3.1.3.1. Zoom

Con el factor de zoom máximo activado y una imagen dedicada a la visualización del mapa de 650x650 píxeles donde cada píxel representa 30x30 metros ésta se corresponde con un área aproximada de 20Km cuadrados cumpliendo así uno de los requerimientos del usuario.

Otro requerimiento del usuario consiste en que la aplicación muestre como vista inicial una vista general del modelo digital de terreno del qué dispone. Para mostrar la visualización inicial del mapa se ha tomado una muestra de cada 12 del fichero que almacena las cotas. A medida que el usuario vaya aumentando el factor de zoom utilizado se tomaran más muestras acorde con la visualización seleccionada para que la imagen mostrada siempre sea del mismo tamaño.

Este proceso es transparente al usuario. El usuario sólo debe activar la herramienta zoom y seleccionar con el ratón el área que desea ampliar sobre el mapa. La herramienta zoom recoge los eventos Mouse Up y Mouse Down del usuario sobre la imagen central y después de realizar los cambios de coordenadas pertinentes llama al método de la clase DirectSeek ExtractAltitudesFromFile().

Una vez obtenido el objeto VisualComponent se procesa de nuevo y se muestra por pantalla el bitmap resultante.

3.1.3.2. Pan

La funcionalidad de pan del visor recae sobre los botones direccionales que rodean a la imagen central y en la función Vertex(). Esta función se encarga de actualizar el vértice que se pasa como parámetro al método ExtractAltitudesFromFile(). De nuevo, los datos extraídos del fichero son volcados en un objeto de la clase VisualComponent para proceder a su transformación a una imagen bitmap.

3.1.3.3. Altitude Information

Esta funcionalidad del módulo DEM va ligada al evento Mouse Hover de la imagen central. Este evento se produce cuando el usuario desplaza el ratón sobre el mapa y nos indica la posición del cursor sobre el mismo tanto en coordenadas UTM como en coordenadas geográficas

Este dato junto con el resto de variables almacenadas en el objeto VisualComponet (que almacena los datos representados en el bitmap) permite obtener las coordenadas UTM del cursor. Una vez determinada la coordenada UTM que ocupa el cursor sobre el mapa se accede al módulo DEM y se muestra por pantalla la altura del punto.

3.1.3.4. Escala y leyenda

Para dotar al mapa de una escala se ha optado por incluir una imagen fija con una longitud en píxel determinada que muestra a su lado derecho el valor en metros que representa.

Partiendo del valor en metros que representa cada píxel en la vista actual (Variable CellSize del objeto VisualComponent) multiplicando este valor por la longitud en píxeles de la escala creada. La imagen utilizada como escala se muestra en la figura 3.6.

Fig. 3.6 Escala

A la izquierda del área dedicada a la visualización del mapa se encuentra la leyenda. Ésta relaciona los distintos colores empleados en el mapa con las alturas de las cotas.

3.2. Implementación de los cambios entre coordenadas

Como se ha comentado en el apartado 1.3 la aplicación debe trabajar con distintos tipos de coordenadas simultáneamente. Éstos son tres:

- coordenadas UTM, usadas por el módulo DEM.
- coordenadas geográficas usadas en el plan de vuelo.
- coordenadas gráficas (las de los objetos dibujados sobre la pantalla del editor).

En los siguientes apartados se describe como se han implementado las proyecciones entre los distintos sistemas de coordenadas que utiliza la aplicación.

3.2.1. Cambio entre coordenadas UTM y coordenadas geográficas

Los algoritmos utilizados para realizar el cambio de coordenadas están basados en las fórmulas de Coticchia-Surace [5]. Dichas ecuaciones fueron planteadas por Alberto Coticchia y Luciano Surace en el *"Bolletino di Geodesia e Science Affini"*, Num. 1.

La precisión que se obtiene ronda el centímetro cuando se utilizan suficientes decimales. En consecuencia, era imperativo que a la hora de programar se utilizaran variables de coma flotante y doble precisión.

Los parámetros del datum ED50 y del elipsoide de Hayford finalmente utilizados para la conversión se recogen en la tabla 3.3.

Tabla 3.3. Datos elipsoide de Hayford y European datum - 50

Semieje mayor (a)	6378388
Semieje menor (b)	6356915.946130
Aplanamiento = a - b / a	297
Longitud punto fundamental	13º 03' 58.741" E
Latitud punto fundamental	52º 22' 51.446" N
Xi	3.36
Eta	1.78

Para almacenar los distintos tipos de coordenadas y realizar las conversiones entre ellas se han diseñado dos clases específicas que representan a ambos sistemas y contienen los algoritmos para permitir las proyecciones. Sus diagramas de clases se presentan en las figuras 3.6 y 3.7.

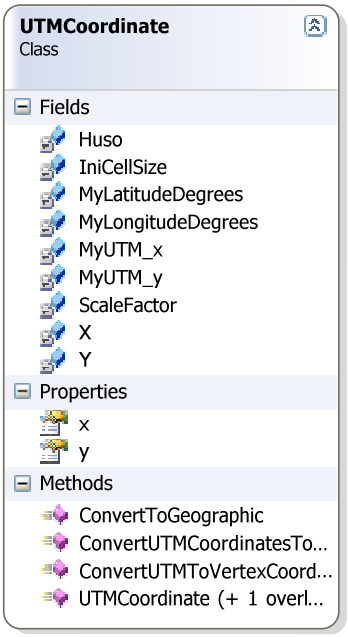


Fig. 3.6 Clase UTMCoordinate

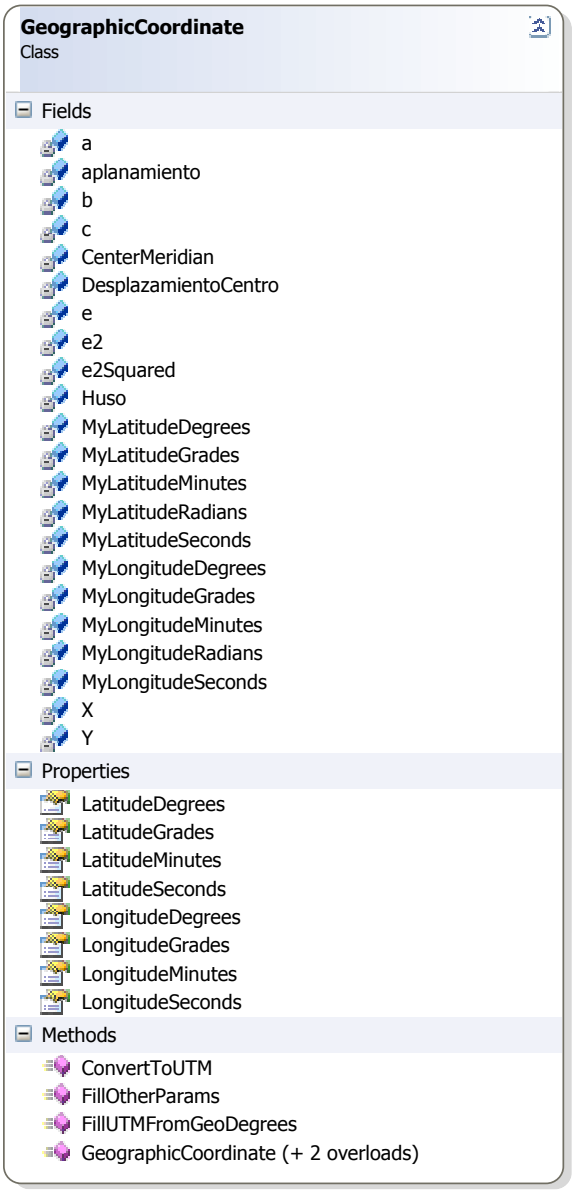


Fig. 3.7 Clase GeographicCoordinate

Los métodos que se encargan de las conversiones entre coordenadas UTM y coordenadas geográficas están contenidos en métodos de las clases respectivas. Para convertir una coordenada UTM a una geográfica se llama al método ConvertToGeographic() de la clase UTMCoordinate. De la misma manera para convertir una coordenada geográfica a una UTM se llama al método ConvertToUTM() presente en la clase GeoCoordinate.

3.2.2. Cambio entre coordenadas UTM y coordenadas gráficas

Cuando un waypoint ya almacenado en coordenadas geográficas debe ser mostrado sobre el mapa se realiza el cambio inverso. Es decir, se convierte a su valor proyectado en UTM y posteriormente a sus coordenadas gráficas.

Dada la funcionalidad de pan y zoom de la aplicación, la posición del ratón sobre el mapa o las coordenadas de los objetos sobre la pantalla se corresponden con una coordenada UTM específica dependiendo del vértice de la imagen mostrada así como del factor de zoom utilizado para mostrarla.

De esta transformación lineal entre las coordenadas de la pantalla del editor y las coordenadas UTM del dato visualizado se encarga el método ConvertUTMCoordinatesToGraphicPoint().

El proceso es bastante simple en este caso :

Se convierten las coordenadas geográficas a UTM. Cuando tenemos las coordenadas expresadas en UTM se divide cada coordenada entre la resolución en metros de la vista que se está mostrando (Variable Cell Size) para obtener las coordenadas referenciadas a la resolución del modelo de elevación del terreno.

Como el modelo cartográfico tiene como vértice inicial del fichero la coordenada UTM (257995, 4752005) antes de dividir entre el valor de CellSize habrá que restarle la coordenada inicial para obtener su valor referenciado al vértice superior izquierdo de la imagen (origen de las coordenadas gráficas).

El cambio entre coordenadas gráficas a geográficas se realiza cuando la aplicación debe mostrar las coordenadas geográficas de los objetos introducidos por el usuario sobre la pantalla o al introducir los waypoints en el plan de vuelo. El método ConvertGraphicPointTo-UTMCoordinate() realiza la transformación inversa a la expuesta con anterioridad.

3.2.3. Test y resultados conversión de coordenadas

Para verificar la precisión con la que se ha implementado la conversión de coordenadas se han llevado a cabo dos test.

Test I . Cambio de coordenadas geográficas a UTM

Partimos de las coordenadas geográficas E 3º 15’ 45” N 42º 14’ 15”. Estas coordenadas se han convertido a UTM con ayuda del conversor de coordenadas que el Institut Cartogràfic de Catalunya ofrece en su web [4].

Los resultados obtenidos se muestran en la figura 3.8.

Fig. 3.8 Resultados ofrecidos por conversor web ICC

La tabla 3.4 muestra los resultados obtenidos así como las diferencias con los resultados del ICC.

Tabla 3.4. Resultados test cambio coordenadas de geográficas a UTM.

	Easting	Northing
Web ICC	521659,5	4676262,6
Coticchia-Surace	521659.59228967712	4676267.0909840865
Diferencia	0.092 metros	4.49 metros

Se ha utilizado el conversor del ICC para realizar el cambio inverso. La tabla 3.5 muestra los resultados convirtiendo las coordenadas UTM obtenidas a coordenadas geográficas:

Tabla 3.5. Resultados test cambio coordenadas de UTM a geográficas

	Coordenadas	Editor .NET
Grados Latitud	42	42.0
Minutos Latitud	14	14.0
Segundos Latitud	15	15.001913751527809
Grados Longitud	3	3.0
Minutos Longitud	15	15.0
Segundos Longitud	45	44.999990334492246

Para calcular cuál es el error cometido en metros debemos considerar que la distancia que representa en el terreno un segundo de latitud varía muy poco si nos movemos del Ecuador hacia los polos, porque los meridianos tienen todos aproximadamente el mismo largo. En el caso de la longitud un segundo representa distintas distancias sobre la tierra, según la latitud en que nos encontremos, debido a que la circunferencia de los paralelos se hace menor al alejarse del Ecuador hasta volverse nula en los polos.

Test II . Cambio de coordenadas geográficas a gráficas

Las coordenadas del primer test no se han escogido al azar. Ante la imposibilidad de georreferenciar ningún punto contenido en el mapa se ha tomado como referencia una fotografía de Google Earth y se ha enfocado sobre el Cap de Creus punto más al oeste de la Península Ibérica. Se ha tomado el punto más al sur de una pequeña península dada su peculiaridad y fácil localización. Sus coordenadas geograficas són E 3º 15’ 45” N 42º 14’ 15”.

La figura 3.9 muestra la localización del punto seleccionado para realizar los test.

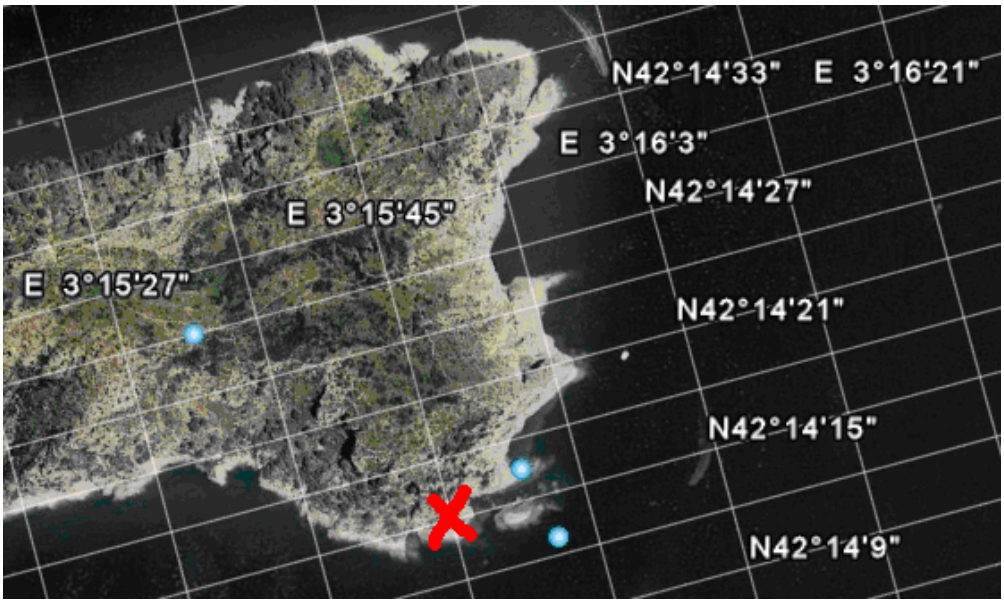


Fig. 3.9 Localización punto test coordenadas

Añadiendo un waypoint sobre la aplicación con las mismas coordenadas geográficas se obtiene la visualización del punto mostrado en la figura 3.10 esta vez sobre el editor implementado.

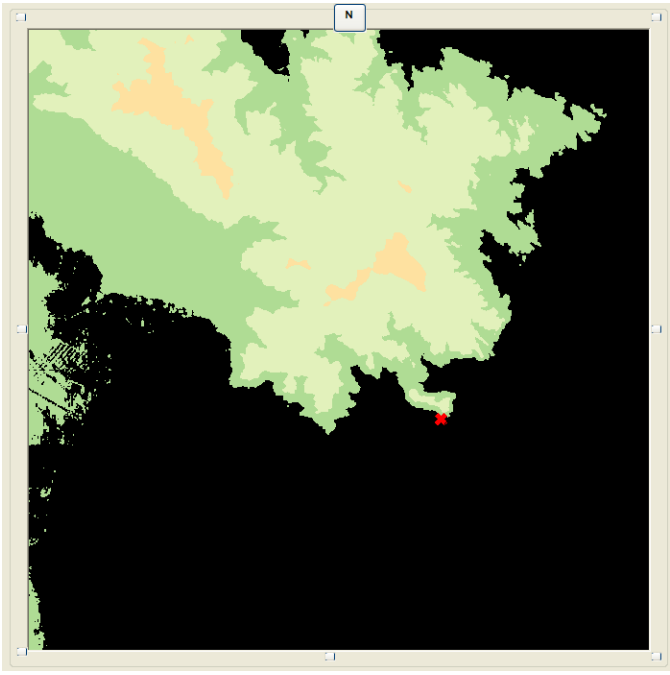


Fig. 3.10 Localización punto sobre editor .NET

En ambos casos el error cometido es inferior a la mitad de la resolución de los datos aportados (15 metros). Recordemos que cada píxel representa una cuadrícula de 30 metros cuadrados de manera que la posición del avión aún cometiendo un error de 5 metros en su coordenada northing se corresponde con la misma posición en la rejilla creada por la proyección UTM.

3.3. Implementación del modelo del plan de vuelo

El modelo de plan de vuelo que finalmente se ha implementado contiene las siguientes clases. Todos los atributos mostrados en las siguientes tablas son editables y modificables utilizando los controles presentes en la aplicación.

3.3.1. Clase WayPoint

El diagrama de clases resultante para la entidad waypoint se presenta en la figura 3.11. Sus principales atributos se recogen en la tabla 3.6.

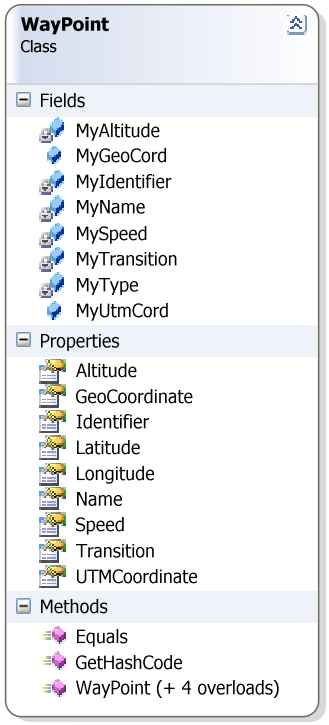


Fig. 3.11 Clase WayPoint

Tabla 3.6. Atributos y métodos de la Clase WayPoint

Name	Utilizado para otorgar un nombre o una breve descripción al waypoint contenido en el plan de vuelo.
Identifier	Utilizado internamente por la aplicación para identificar al waypoint de manera unívoca. Ej. Bcn[1]
GeoCoordinate	Contiene las coordenadas geográficas del waypoint.
UTMCoordinate	Contiene las coordenadas UTM del waypoint.
Type	Campo utilizado para definir el tipo de waypoint.
Transition	Los posibles valores para este campo son fly-by y fly-over.
Speed	Contiene el valor de velocidad para sobrevolar el waypoint.
Altitude	Contiene la altitud a la que se debe sobrevolar el waypoint.

3.3.2. Clase Leg

Todas las clases que se han implementado para definir los distintos tipos de legs heredan de esta clase base. Sus principales atributos se recogen en la tabla 3.7:

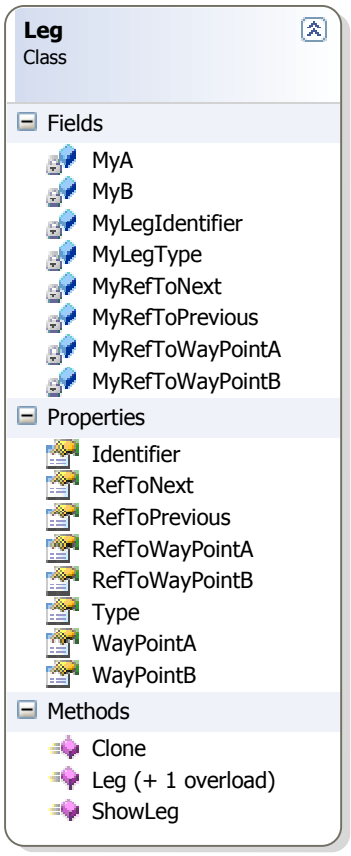


Fig. 3.12 Clase Leg

Tabla 3.7. Atributos y métodos de la Clase Leg

Identifier	Campo que identifica al leg. Suma literal de abreviatura del tipo y de los identificadores de los waypoints que une.
RefToNext	String que contiene el identificador del leg siguiente
RefToPrevious	String que contiene el identificador del leg previo
RefToWayPointA	String que contiene el identificador del waypoint origen
RefToWayPointB	String que contiene el identificador del waypoint destino.
WayPointA	Contiene los datos del waypoint origen del leg.
WayPointB	Contiene los datos del waypoint destino del leg.
Type	Utilizado temporalmente para guardar el tipo de leg que el usuario desea crear y mostrarlo por pantalla antes de su creación.

En los siguientes apartados se muestran las clases desarrolladas para representar los diferentes tipos de legs. Estas clases al heredar de la clase base contienen todos los campos de la misma además de algunos atributos adicionales.

Como se puede observar todas las clases incluyen el método ShowLeg() que se encarga de mostrar el leg por pantalla. Este método ha sido reescrito para mostrar cada tipo de leg de una manera diferente visualmente.

3.3.2.1. Leg Track to Fix

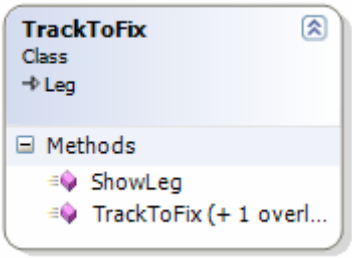


Fig. 3.13 Clase Leg Track to Fix

No se incluye ningún otro atributo adicional respecto a la clase base para definir la trayectoria de este leg ya que se trata de una línea recta entre los dos waypoints que contiene.

3.3.2.2. Leg Radius To Fix

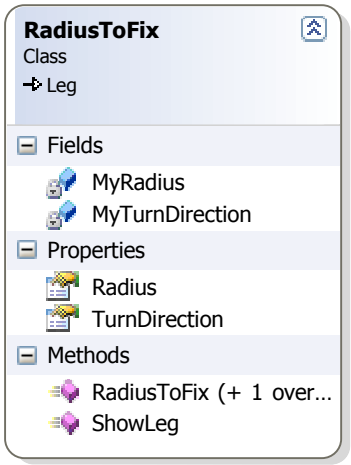


Fig. 3.14 Clase RadiusToFix

La trayectoria que se corresponde con este leg es un arco definido entre los dos waypoint que contiene y el radio del círculo que describe el UAV. Esta

trayectoria queda definida por el centro del giro y los parámetros Radius (radio) y TurnDirection (sentido de giro). Entendiendo Radius como el radio del círculo que el avión debe trazar entre los dos waypoints.

El sentido de giro es una variable string que puede tomar dos valores : clockwise si es giro se produce en el sentido de giro de las agujas del reloj y unclockwise si el giro se produce en sentido contrario.

3.3.2.3. Leg Direct to Fix

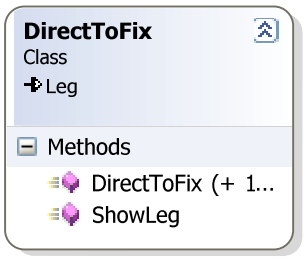


Fig. 3.15 Clase Leg Direct to Fix

Al igual que el leg Track to fix define una trayectoria en línea recta hasta el waypoint destino pero en este caso independientemente de la posición del avión. Se aconseja su uso inmediatamente después de un leg Hold Pattern o cuando se quiera dirigir al avión a una posición dada sin considerar la posición actual del mismo.

3.3.2.4. Leg Hold Pattern

El diagrama de clases de este tipo de leg se presenta en la figura 3.16. Éste define una trayectoria elíptica sobrevolando el waypoint destino del leg con los atributos presentados en la tabla 3.8.

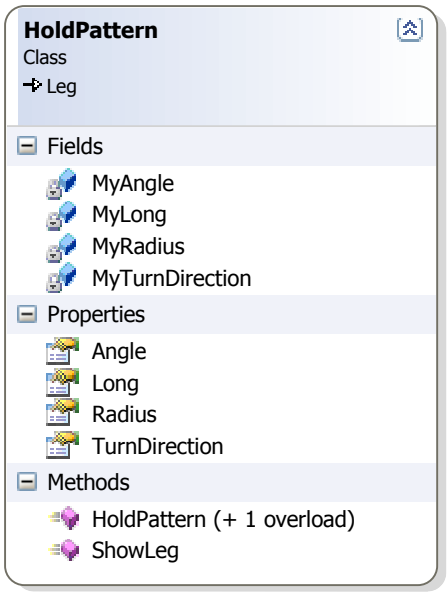


Fig. 3.16 Clase Leg HoldPattern

Tabla 3.8. Atributos y métodos de la Clase HoldPattern

Angle	Utilizado en RNAV define la manera en que el UAV debe aproximarse al waypoint destino.
Long	Define la longitud del segmento recto del patrón.
Radius	Define el arco de giro del segmento semicircular del patrón.
TurnDirection	Del mismo modo que en el leg radius to fix el sentido de giro puede tomar los valores clockwise y unclockwise.

3.3.2.5. Leg Intersección

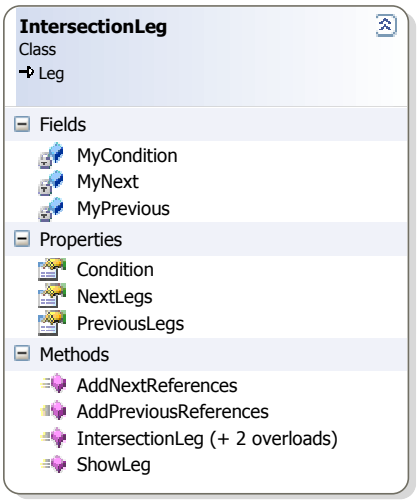


Fig. 3.17 Clase IntersectionLeg

Se ha comentado anteriormente (véase el apartado 2.3.2.3) que un leg intersección debe establecerse en cualquier punto donde dos trayectorias se unen en un mismo waypoint.

Uno de los inconvenientes de considerar las intersecciones de trayectorias como tipos de legs es que las referencias a los legs siguientes y previos de la intersección no pueden ser definidas antes de la creación de la misma.Es decir, si a una trayectoria compuesta por legs simples añadimos una intersección las referencias establecidas con anterioridad deben ser de nuevo redefinidas. Los métodos que se encargan de introducir las referencias a los legs previos y siguientes cuando el usuario crea una intersección son AddNextReferences() y AddPreviousReferences().

Hacer notar que en este caso los atributos de la clase base Leg RefToNext y RefToPrevious se utilizan para indicar que leg ha sido asignado por defecto.

Tabla 3.9. Atributos y métodos de la Clase IntersectionLeg

RefToNext	String que contiene el identificador del leg que seguirá la intersección de no cumplirse ninguna condición.
RefToPrevious	Análogo a RefToNext y usado en el modo reverso.
Condition	String que enlaza con las condiciones de la intersección.
NextLegs[]	ArrayList que contiene todos los posibles legs siguientes a la intersección. Es decir, todas las trayectorias que parten de ella.
PreviousLegs[]	ArrayList que contiene todos los posibles legs previos a la intersección. Es decir, todas las trayectorias que conducen a ella.

3.3.3. Clase Stage

En una misma fase quedarán recogidos aquellos legs que persigan un propósito común. Un leg inicial de una fase será aquel que no tenga ninguna trayectoria anterior. De manera análoga un leg final será aquel que no tenga ningún leg siguiente.

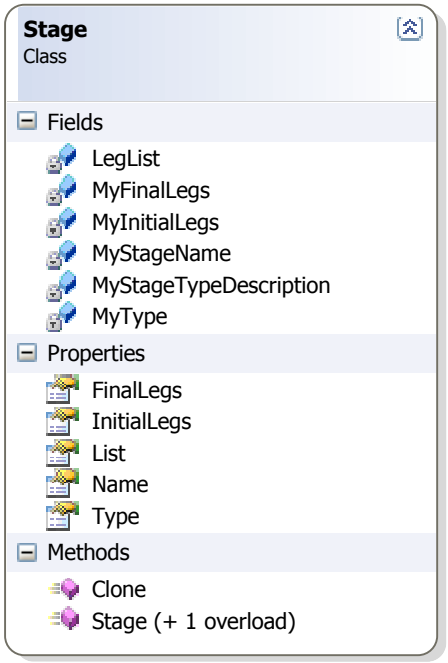


Fig. 3.18 Clase Stage

Tabla 3.10. Atributos y métodos de la Clase Stage

Name	Campo string que indica el nombre de la fase.
Type	Campo string que indica el tipo de la fase. Los posibles valores para este campo han sido expuestos en el apartado 3.3.2.2.
StageTypeDescription	Aporta la descripción del tipo de fase.
LegList	ArrayList que contiene todos los legs de la fase.
FinalLegs	ArrayList que contiene los legs iniciales de la fase.
InitialLegs	ArrayList que contiene los legs finales de la fase.

3.3.4. Clase SimulatedFlight

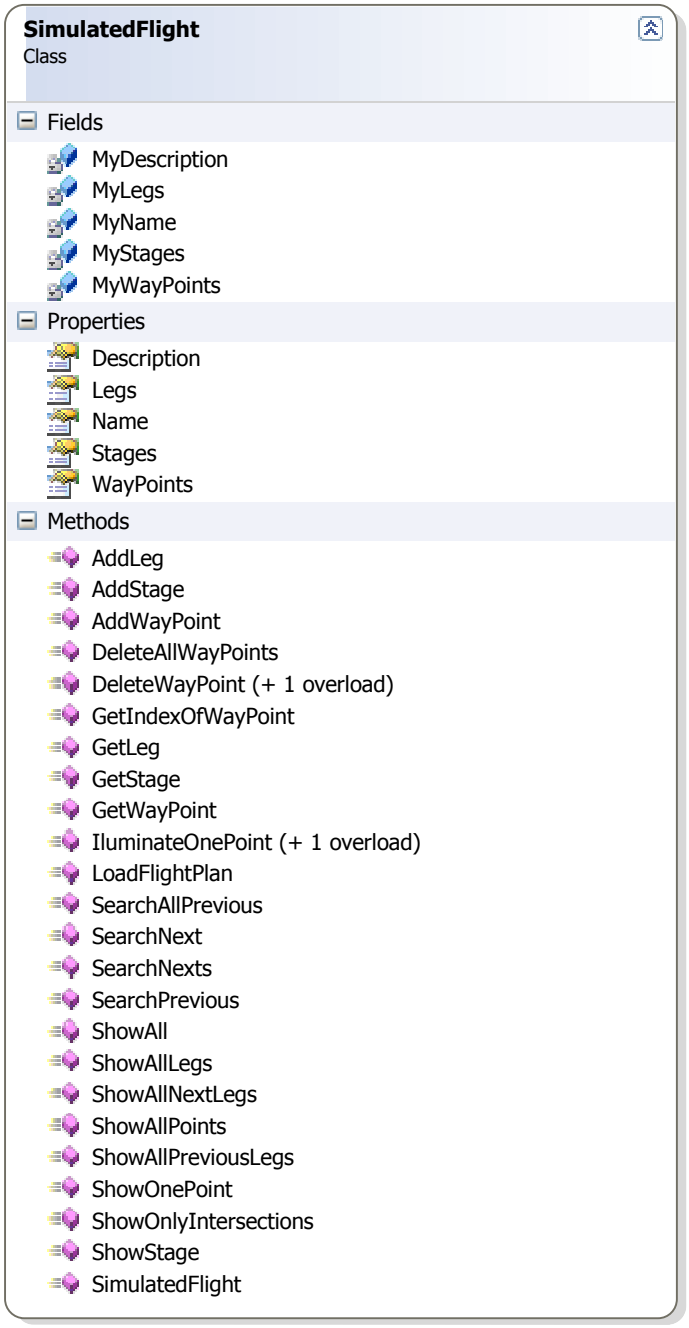


Fig. 3.19 Clase SimulatedFlight

La clase SimulatedFlight contiene tres ArrayList denominadas Legs, WayPoints y Stages. Cada una de estas colecciones contiene elementos de las clases Legs, Waypoints y Stages respectivamente.

El ArrayList de WayPoints almacena todos y cada uno de los waypoints introducidos por el usuario en el plan de vuelo. El ArrayList de Legs almacena temporalmente los legs introducidos hasta que estos son guardados en una fase del plan de vuelo. Éstos son almacenados en el ArrayList Stages con el resto de parámetros que definen a la fase. La información contenida finalmente en el ArrayList Stages junto con el resto de los parámetros del plan de vuelo contiene toda la información necesaria para reproducir el plan de vuelo. La tabla 3.11 recoge estos aspectos.

Tabla 3.11. Atributos de la clase SimulatedFlight

Name	Campo String que almacena el nombre del plan de vuelo.
Description	Campo String que permite almacenar una breve descripción del plan de vuelo.
WayPoints []	Colección que almacena todos los waypoints contenidos en el plan de vuelo.
Legs[]	Colección que almacena los legs incluidos en el plan de vuelo por el usuario y no han sido almacenados en ninguna fase.
Stages[]	Colección que almacena los datos de todas las fases del plan de vuelo editado.

Algunos de los métodos contenidos en esta clase tienen como finalidad gestionar los objetos que componen el plan de vuelo. Añadir o eliminar waypoints, legs o fases del vuelo. También muestra por pantalla determinados grupos de objetos que componen el plan de vuelo.

Tabla 3.12. Métodos de la clase SimulatedFlight

AddWayPoint	Añade un waypoint a la colección WayPoints
AddLeg	Añade un leg a la colección Legs
AddStage	Añade una fase a la colección Stages
DeleteWayPoint	Elimina un waypoint determinado.
DeleteAllWayPoints	Elimina todos los waypoints.
ShowAll	Muestra en la vista seleccionada los waypoints y los legs.
ShowAllLegs	Muestra todos los legs que componen la fase seleccionada.
ShowStage	Muestra una determinada fase del plan de vuelo.
ShowOnePoint	Muestra un determinado waypoint.
ShowOnlyIntersections	Muestra únicamente las intersecciones de una fase
ShowAllNextLegs	Muestra las posibles siguientes trayectorias de un leg determinado que se pasa como parámetro.
ShowAllPreviousLegs	Muestra las posibles anteriores trayectorias de un leg determinado que se pasa como parámetro.
ShowAllPoints	Muestra todos los waypoints contenidos.
IluminateOnePoint	Muestra un waypoint como seleccionado.

3.4. Implementación interfaz de usuario

Dadas las posibilidades de .NET para crear interficies gráficas complejas de usuario y su capacidad para desarrollar aplicaciones Windows estándar ha resultado una herramienta excelente para desarrollar el presente editor.

3.4.1. Herramientas

Para implementar la interfaz de usuario se han empleado varios espacios de nombres que ofrece .NET. Un espacio de nombre no es más que un conjunto organizado de clases ya implementadas. Entre todos ellos cabe destacar el namespace Forms. El espacio de nombres System.Windows.Forms dispone de diferentes clases de controles que se pueden utilizar para crear interficies de usuario.

Controles

Un control es una clase con aspecto visual. Algunos controles están diseñados para la entrada de datos en la aplicación, por ejemplo, los controles TextBox y ComboBox. Otros controles muestran datos de la aplicación, por ejemplo, Label y ListView. El espacio de nombres también dispone de controles para invocar comandos en la aplicación, por ejemplo, Button.

Los distintos controles añadidos tienen varias funciones:

- Para crear nuevos objetos del plan de vuelo o eliminarlos.
- Para mostrar la información de los objetos seleccionados.
- Para establecer relaciones entre las clases del plan de vuelo.
- Para verificar las relaciones establecidas.

Cada uno de los controles ofrece información contextual sobre su función ayudando al usuario en el manejo de la aplicación. Para ello se ha utilizado la propiedad ToolTip presente en casa uno de ellos.

Eventos

La aplicación captura las acciones del usuario mediante el modelo de eventos que ofrece .NET. Los eventos son el mecanismo que tienen los objetos para avisarnos de algún cambio en su estado y dejarnos la posibilidad de actuar frente al cambio ejecutando cierto código.

En la creación de interfícies gráficas los eventos cobran gran importancia ya que la ejecución del código se hará en la mayoría de los casos a raíz de los eventos que produce el usuario al interactuar con los controles de la aplicación.

Un breve resumen de los controles, eventos y namespaces utilizados para la implementación se recoge en el Anexo A.

Sin embargo el control de eventos que proporciona .NET no es suficiente para gestionar los eventos que produce el usuario sobre los objetos dibujados sobre

la imagen central. Estos no son objetos que .NET reconozca como propios de manera que eventos como MouseClick, Mouse Hover. etc. no pueden ser aplicados a los objetos contenidos en el plan de vuelo.

Para poder seleccionar los objetos una vez dibujados sobre el mapa se ha utilizado la técnica conocida como *picking*. Esta localiza los objetos ya dibujados sobre el mapa según su posición y realiza una búsqueda sobre los objetos almacenados en las distintas ArrayList del plan de vuelo.

Una vez determinado que objeto es el que se quiere pulsar éste es seleccionado para su edición.

3.4.2. Descripción de la interfaz

La aplicación consta de un solo formulario y el resto de controles que permiten al usuario introducir datos y comandos a la aplicación están contenidos en él. El aspecto de la interfaz de usuario se muestra en la figura 3.20.

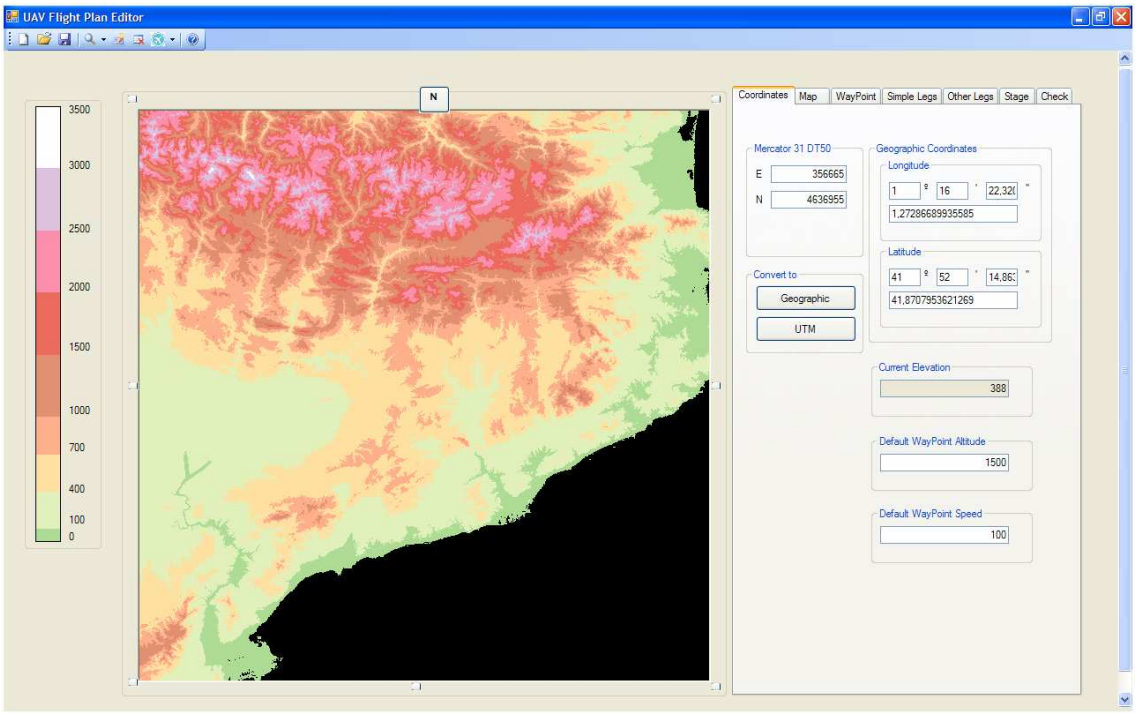


Fig. 3.20 Aspecto aplicación

El formulario se compone de:

- Una imagen central donde se muestra el mapa creado a partir de los datos del modelo digital del terreno.

- Leyenda

- Botones direccionales encargados de dirigir a la herramienta pan distribuidos alrededor de la imagen central.

- Barra de herramientas que da acceso entre otras funciones a cargar o grabar un plan de vuelo.

- Agrupación de pestañas (TabControl) donde se recogen la mayoría de funcionalidades del editor.

Por simplicidad se han agrupado todos los controles de visualización en un único control que contiene varias pestañas. Éste cambia su aspecto según la acción que se desee emprender. Se han intentado agrupar los controles en pestañas según el objeto que se desee tratar del plan de vuelo para hacer al usuario su uso algo más intuitivo evitando la creación constante de múltiples formularios. Así los nombres de las pestañas se corresponden simplemente con los objetos a tratar. Las distintas pestañas contenidas y sus funcionalidades se enumeran a continuación:

- Coordinates: Pestaña que muestra las coordenadas del punto seleccionado tanto en coordenadas UTM como en coordenadas geográficas según la posición que ocupe el cursor sobre la imagen central.

- Map: Contiene un visor reducido del mapa donde se muestra la vista seleccionada y la escala utilizada por la vista principal.

- WayPoint: Permite seleccionar y añadir waypoints así como editar sus características. También contienen un listado de todos los waypoints incluidos en el plan de vuelo que se está editando.

- Simple Leg: Permite crear legs a partir de dos waypoints previamente seleccionados y editar sus características

- Other Legs: Pestaña que permite editar los legs del tipo intersección, establecer que condición debe ser considerada al llegar a ellos y modificar el camino que se tomara por defecto.

- Stage: Permite agrupar y grabar el trabajo realizado hasta el momento en una fase del plan de vuelo. También permite validar la fase en la que se está trabajando.

- Check: Pestaña que permite verificar de una manera visual el plan de vuelo al completo y comprobar que las referencias entre los objetos que lo componen sean consistentes. Para ello, partiendo de cada uno de los legs iniciales va iluminando los legs referenciados como siguientes. Al llegar a un leg del tipo intersección sigue el camino que se ha marcado como default.

Para más información sobre el uso de estas funcionalidades consulte el manual adjunto en el anexo B.

3.5. Generación del fichero XML

El propósito de la aplicación es cargar el plan de vuelo editado en el UAV de manera que este sea leído progresivamente por el piloto automático.

Para generar el fichero XML la aplicación utiliza el espacio de nombres System.Xml.Serialization. Éste contiene un conjunto de clases que se utilizan para serializar objetos es decir, convertir los mismos en archivos XML. (Véase en anexo A para más información sobre su funcionamiento).

La información que la aplicación serializa es el contenido en la clase SimulatedFlight ya que contiene toda la información necesaria para reproducir el plan de vuelo editado. Para acceder a esta funcionalidad el usuario sólo debe grabar el plan de vuelo pulsando el botón grabar accesible desde la barra de herramientas.

Como muestra del proceso de serialización se adjunta el fichero generado por la aplicación para un plan de vuelo que contiene una única fase. La fase consta de un leg track to fix, un leg radius to fix, un leg intersección y dos legs finales (del tipo direct y track to fix). Los legs que componen la fase se muestran en la figura 3.21. Para acortar la extensión del fichero XML únicamente se muestra el detalle de la información del primer leg de la lista y su waypoint de origen en el listado que sigue a la figura.

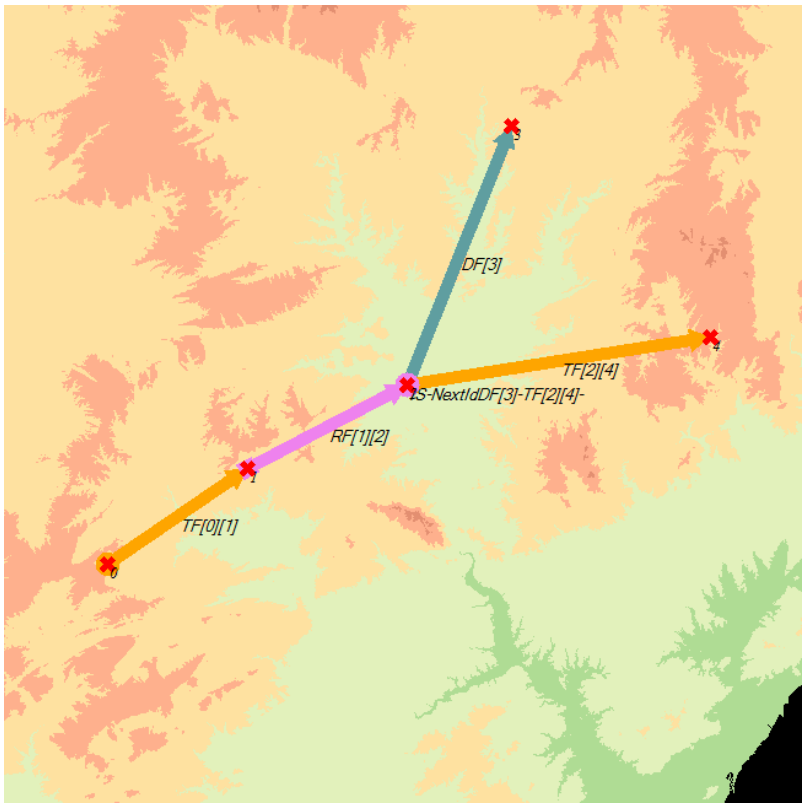


Fig. 3.21 Fase de ejemplo utilizada para generar XML

```
<?xml version="1.0" encoding="utf-8" ?>
<SimulatedFlight xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Name>Sample XML</Name>
<Description>Flight Plan Brief Description Text</Description>
<Stages>
<anyType xsi:type="Stage">
<Name>Sample XML Stage</Name>
<Type>Route</Type>
<List>

<anyType xsi:type="TrackToFix">
<WayPointA>
<Altitude>1500</Altitude>
     <GeoCoordinate>
        <LatitudeDegrees>41.557214038517287</LatitudeDegrees>
        <LongitudeDegrees>1.4542705563319514</LongitudeDegrees>
     </GeoCoordinate>
     <UTMCoordinate>
     <x>371095</x>
     <y>4601855</y>
     </UTMCoordinate>
  <Speed>100</Speed>
  <Name />
  <Transition>Fly - By</Transition>
  <Identifier>[0]</Identifier>
</WayPointA>

     <WayPointB>
     [...]
           </WayPointB>

  <Type>TF</Type>
  <Identifier>TF[0][1]</Identifier>
  <RefToNext>RF[1][2]</RefToNext>
  <RefToPrevious>Initial Leg</RefToPrevious>
  <RefToWayPointA>[0]</RefToWayPointA>
  <RefToWayPointB>[1]</RefToWayPointB>
    </anyType>
+ <anyType xsi:type="RadiusToFix">
+ <anyType xsi:type="DirectToFix">
+ <anyType xsi:type="TrackToFix">
+ <anyType xsi:type="IntersectionLeg">
     </List>
<InitialLegs>
  <anyType xsi:type="xsd:string">TF[0][1]</anyType>
</InitialLegs>
<FinalLegs>
  <anyType xsi:type="xsd:string">DF[3]</anyType>
  <anyType xsi:type="xsd:string">TF[2][4]</anyType>
 </FinalLegs>
 </anyType>
</Stages>
</SimulatedFlight>
```

4. CONCLUSIONES

Este capítulo realiza un repaso a los problemas que han aparecido a lo largo de la realización del proyecto, seguido de propuestas sobre posibles mejoras a realizar en la aplicación. Finaliza con una valoración del impacto ambiental y personal del mismo.

El resultado del trabajo es una aplicación Windows que permite editar un plan de vuelo y definir trayectorias complejas basadas en conceptos del sistema de navegación de área.

Durante todo el proceso el operador de tierra es asistido tanto para la edición del plan de vuelo como para visualizar los posibles obstáculos presentes en el terreno mediante la incorporación de un modelo digital de elevaciones.

Durante el desarrollo del proyecto se han analizado los diferentes sistemas de coordenadas que intervienen en la aplicación y se han implementado las diferentes proyecciones necesarias para realizar los cambios entre los mismos.

4.1. Líneas futuras

Aunque la aplicación permite definir las trayectorias básicas para formar un plan de vuelo completo sería aconsejable definir tipos de legs que permitan establecer otros patrones más complejos de comportamiento, como por ejemplo, patrones de escaneo de área.

Definir con más profundidad las condiciones que soportará el sistema y que se tendrán cuenta al llegar a un leg intersección.

Aunque estas condiciones puedan usarse para lograr un comportamiento iterativo en la trayectoria del UAV sería aconsejable introducir grupos de legs donde se pudieran definir estas iteraciones con mayor facilidad.

La interficie gráfica del presente editor puede convertirse en la estación de tierra y lograr así monitorizar la posición del UAV durante el vuelo y verificar que éste se ajusta al definido.

4.2. Impacto ambiental

Hoy en día, es importante considerar el posible impacto ambiental que puede tener la implementación de cualquier proyecto tecnológico.

La realización del presente proyecto no repercute en si misma a dañar el medio ambiente ya que se trata del diseño de un software. La aplicación está pensada para un UAV que al ser un vehículo aéreo no tripulado es más ligero comparado con un avión tripulado lo que permite un menor gasto de energía en la operación y de materias primas en la fabricación del aparato. Además, uno de los objetivos del proyecto global donde se enmarca este software es muy útil para ayudar a preservar el medio ambiente, ya que una posible misión del

avión es la de informar visualmente sobre el foco y progreso de incendios. Esta ayuda favorecería la intervención de las autoridades en el lugar exacto del incendio y permitiría una extinción del fuego más rápida y controlarlo de forma más eficiente.

4.3. Valoración personal

El proyecto ha supuesto la asimilación de una gran cantidad de conocimientos en bastantes áreas. Aunque basado en una única tecnología se ha desarrollado en un entorno multidisciplinar enriqueciendo mis conocimientos en áreas desconocidas hasta el momento.

La proyección UTM implementada era una curiosidad histórica por resolver en la que hasta ahora no me había detenido con tanto empeño e interés.

La complejidad de la interfaz de usuario me ha permitido profundizar en aspectos desconocidos de la tecnología .NET.

En conclusión, la valoración personal sobre el proyecto es muy positiva, ya no sólo por el resultado obtenido, sino, básicamente por la gran cantidad de conocimientos que me ha aportado.

5. BIBLIOGRAFÍA

[1] Guidance Material for the Design of Terminal Procedures for Area Navigation (DME/DME, B-GNSS, Baro-VNAV & RNP-RNAV) EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION. EUROPEAN AIR TRAFFIC MANAGEMENT PROGRAMME

[2] Ignacio Alonso Fernández-Copel . “Localizaciones Geográficas. Las Coordenadas Geográficas y la Proyección UTM”. Departamento de Ingeniería Agrícola y Forestal Escuela Técnica Superior de Ingenierías Agrarias. Palencia. Universidad de Valladolid.

[3] www.wikipedia.org

[4] www.icc.cat/web/content/php/geotex/geoutm.php

[5] www.gabrielortiz.com

[6] http://www.fomento.es/MFOM/LANG_CASTELLANO/DIRECCIONES_-GENERALES/AVIACION_CIVIL/PROGRAMAS/NAVEGACION_AREA/


UPC
epsc
Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANEXOS

ANEXO A. Entorno .NET

A.1 Entorno .NET

Microsoft. NET es un conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando en los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir software en forma de servicios que puedan ser suministrados remotamente y que puedan combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados.

La idea principal que persigue la plataforma .NET es que el código sea 100% portable, es decir que una aplicación escrita bajo este entorno se ejecutará en cualquier dispositivo sin tener que hacer ningún tipo de adaptación o modificación en el código.

Dadas las posibilidades de .NET para crear interficies gráficas complejas de usuario y su capacidad para desarrollar aplicaciones Windows estándar resulta una herramienta excelente para implementar el presente editor.

A continuación se presenta una breve introducción al entorno repasando sus características y sus librerías más destacadas.

A.1.1. CLR

El Common Language Runtime es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas y a las que ofrece múltiples servicios que simplifican su desarrollo. Podemos decir que es el equivalente a la Máquina Virtual de Java y ejecuta aplicaciones escritas en lenguaje MSIL.

A.1.2. MSIL

Ninguno de los compiladores que generan código integrados en la plataforma .NET produce código máquina para CPU x86 ni para ningún otro tipo de CPU concreta, sino que generan código escrito en el lenguaje intermedio conocido como Microsoft Intermediate Languaje (MSIL). Este es el único código que es capaz de interpretar el CLR.

La principal ventaja de MSIL es que facilita la ejecución multiplataforma y la integración de varios lenguajes. Sin embargo, dado que las CPU no pueden ejecutar directamente instrucciones MSIL antes de ejecutarlo habrá que convertirlo al código nativo de la CPU donde se vaya a ejecutar De esto se encarga una componente del CLR conocida como JIT (Just in Time) o jitter que va convirtiendo dinámicamente el código MSIL a ejecutar en código nativo según sea necesario durante la ejecución.

A.1.3. BCL

La Base Class Library es una librería incluida en .NET Framework formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente usadas a la hora de escribir programas. Esta librería esta escrita en MSIL por lo que puede usarse desde cualquier lenguaje Es tal la riqueza que incluso pueden crearse lenguajes que carezcan de librería de clases propia como es el caso de C#.

A.1.4 Windows Forms

La BCL (Librería de Clases Base) que forma parte de .NET Framework y sobre la que se basan la mayoría de aplicaciones desarrolladas, esta organizada en namespaces. Lo que se conoce como Windows Forms no es otra cosa que el namespace System.Windows.Forms que contiene las clases que se utilizan para crear interficies de usuario basadas en ventanas para aplicaciones estándar.

Hay que remarcar que Windows.Forms no es un simple namespace más, sino que es posiblemente el más importante ya que cualquier aplicación que deba tener una interfaz visual de usuario hará uso de él.

Controles

Un control se puede definir como un componente con aspecto visual. Son en resumen los objetos que forman la interfície gráfica y todos ellos descienden de una misma clase de la cual obtienen un funcionamiento básico y común a todos ellos.

El espacio de nombres **System.Windows.Forms** dispone de diferentes clases de controles que se pueden utilizar para crear interfaces de usuario completas. Algunos controles están diseñados para la entrada de datos en la aplicación, por ejemplo, los controles TextBox y ComboBox. Otros controles muestran datos de la aplicación, por ejemplo, Label y ListView. El espacio de nombres también dispone de controles para invocar comandos en la aplicación, por ejemplo, Button.

Form

Un Form o formulario no es más que un control que representa una ventana genérica propia de una interfaz de usuario. Es la pieza básica en la que se basan las interfaces gráficas ya que todos los demás controles que contenga la aplicación serán mostrados en el interior de cómo mínimo uno de estos controles.

Label

Un Label o etiqueta es un control cuya finalidad es simplemente mostrar un texto dentro de la interfaz gráfica. Las etiquetas son útiles para dar información al usuario y para describir los diferentes elementos de la interfaz. Uno de sus usos más habituales es colocarlo al lado de un TextBox para describir el texto que se ha de introducir en él.

TextBox

Un TextBox o caja de texto es el control más comúnmente utilizado para introducir datos por parte del usuario. Básicamente es un rectángulo en el que pueden caber una o varias líneas de texto.

Button

Un Button o botón es otro de los controles más sencillos de Windows.Forms pero al mismo tiempo uno de los más utilizados. Su finalidad es ser pulsado para que se realice cierta acción como consecuencia de su pulsación.

ComboBox

Representa un control de cuadro combinado de Windows.

CheckBox

Se utiliza un control **CheckBox** para dar al usuario una opción del tipo verdadero/falso o sí/no.

TreeView

Muestra una colección jerárquica de elementos con etiquetas, representado cada uno por un **TreeNode**.

Timer

Implementa un temporizador que provoca un evento en los intervalos definidos por el usuario. Este temporizador está optimizado para su uso en aplicaciones de Windows Forms y se debe utilizar en una ventana.

Menús y barras de herramientas

Windows.Forms contiene un amplio conjunto de clases para que pueda crear sus propias barras de herramientas y menús personalizadas, con un aspecto y comportamiento modernos

Menu

El menú es también un aspecto característico de las aplicaciones gráficas basadas en ventanas. Se corresponde con una clase abstracta y por lo tanto no se puede usar para crear una instancia de ella. Realmente las clases que se utilizan son tres que derivan de ella. La clase MainMenu, ContextMenu y MenuItem.

La clase MainMenu representa el menú principal de una ventana y es la que se sitúa generalmente en la parte superior del formulario.

La clase ContextMenu representa los menús contextuales asociados a un control específico.

La clase MenuItem representa a un elemento que aparece dentro de otro menú. Pueden aparecer tanto dentro del menú principal como de uno Contextual y pueden contener una colección de objetos MenuItem dentro de ellos formando los submenús de este.

Tool Bar

Representa una barra de herramientas de Windows.

Tool Tip

Representa una pequeña ventana emergente rectangular que muestra una breve descripción de la finalidad de un control cuando el usuario sitúa el puntero sobre el control.

ProgressBar

Representa un control de barra de progreso de Windows.

Cuadros de diálogo

OpenFileDialog

Solicita al usuario que abra un archivo.

Save File Dialog

Pregunta al usuario si desea seleccionar una ubicación para guardar un archivo. Esta clase no se puede heredar.

MessageBox

Muestra un cuadro de mensaje que puede contener texto, botones y símbolos que informan e instruyen al usuario.

Eventos importantes de un control

Los eventos son el mecanismo que tienen los objetos para avisarnos de algún cambio en su estado y dejarnos la posibilidad de actuar frente al cambio ejecutando cierto código.

En la creación de interfícies gráficas los eventos cobran gran importancia ya que la ejecución del código se hará en la mayoría de los casos a raíz de los eventos que produce el usuario al interactuar con los controles de la aplicación.

Los eventos más importantes y comunes a la mayoría de controles se describen brevemente a continuación:

Click	Ocurre cuando se hace click sobre el control
Double Click	Ocurre cuando se hace doble click sobre el control
Mouse Move	Ocurre cuando el puntero del ratón se mueve sobre un control.
Mouse Down	Ocurre cuando el puntero del ratón esta sobre el control y se pulsa un botón del mismo.
Mouse Up	Ocurre cuando el puntero del ratón esta sobre el control y se suelta un botón.
TextChanged	Ocurre cuando cambia el valor de la propiedad text del control.

Containers

Un container o contenedor, es un objeto que puede contener cero o más componentes. Nos sirve como mecanismo de agrupación y asegura que un conjunto de componentes serán encapsulados y tratados de una forma similar.

TabControl

Administra un conjunto relacionado de páginas de fichas. En ellas se pueden agrupar fácilmente los controles.

TabPage

Representa una sola página de fichas en un objeto TabControl.

GroupBox

Representa un control Windows que muestra un marco alrededor de un grupo de controles con un título opcional.

A.1.5. Otros namespaces utilizados

Para el desarrollo de la aplicación se han utilizado una gran variedad de namespaces que ofrece .NET. Los más destacados System.Drawing, System.Collections, Systems.Serialization, System.Threading, System.IO, System.ComponentModel y System.Data entre otros. Sigue a continuación una breve descripción de los más relevantes.

A.1.5.1. System.Drawing

El espacio de nombres **System.Drawing** proporciona acceso a funcionalidad de gráficos básica de GDI+. Se ofrece una funcionalidad más avanzada en los espacios de nombres System.Drawing.Drawing2D, System.Drawing.Imaging y System.Drawing.Text.

La clase Graphics proporciona métodos para dibujar en el dispositivo de pantalla. Clases como Rectangle y Point encapsulan primitivos de GDI+. La clase Pen se utiliza para dibujar líneas y curvas, mientras que las clases derivadas de la clase abstracta Brush se utilizan para rellenar el interior de las formas.

A.1.5.2 System.Collections

El espacio de nombres **System.Collections** contiene interfaces y clases que definen varias colecciones de objetos, como listas, colas, matrices de bits, tablas hash y diccionarios.
Permite el uso de ArrayList que representan una colección de objetos no genéricos a los que se puede obtener acceso por índice.

A.1.5.3 System.Xml.Serialization

El espacio de nombres **System.Xml.Serialization** contiene clases que se utilizan para serializar objetos en secuencias o documentos con formato XML.

La clase central en el espacio de nombres es la clase XmlSerializer. Para utilizar esta clase, use el constructor XmlSerializer con el fin de crear una instancia de la clase utilizando el tipo de objeto que se va a serializar. Tras crear XmlSerializer, cree una instancia del objeto que se va a serializar. También hay que crear un objeto para escribir el archivo en un documento o una secuencia, como Stream, TextWriter o XmlWriter. A continuación, hay que llamar al método Serialize para convertir el objeto en un documento XML.

Para deserializar un objeto de un documento XML, hay que crear un objeto adecuado con el fin de leer el documento o la secuencia (de nuevo, Stream, TextWriter o XmlWriter). Posteriormente se invoca al método Deserialize mientras se convierte el objeto resultante en el tipo del objeto original (serializado).

ANEXO B . MANUAL DE LA APLICACIÓN

Tabla de contenidos Manual Aplicación

B.1 Introducción

El propósito de este manual es proveer a los operadores de la aplicación de una guía competa conteniendo toda la información necesaria para editar un plan de vuelo. La aplicación prové un interfaz amigable que permite al operador diseñar el plan de vuelo de una manera sencilla. El plan de vuelo una vez editado puede ser almacenado en formato XML y recuperado posteriormente.

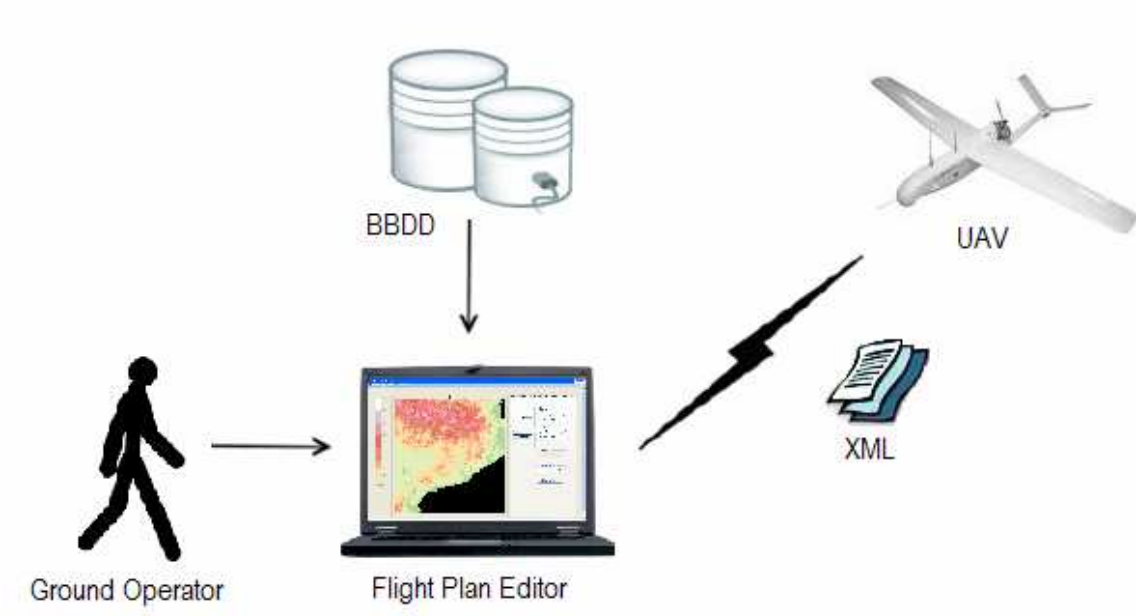


Fig. B.1 Escenario marco de la aplicación

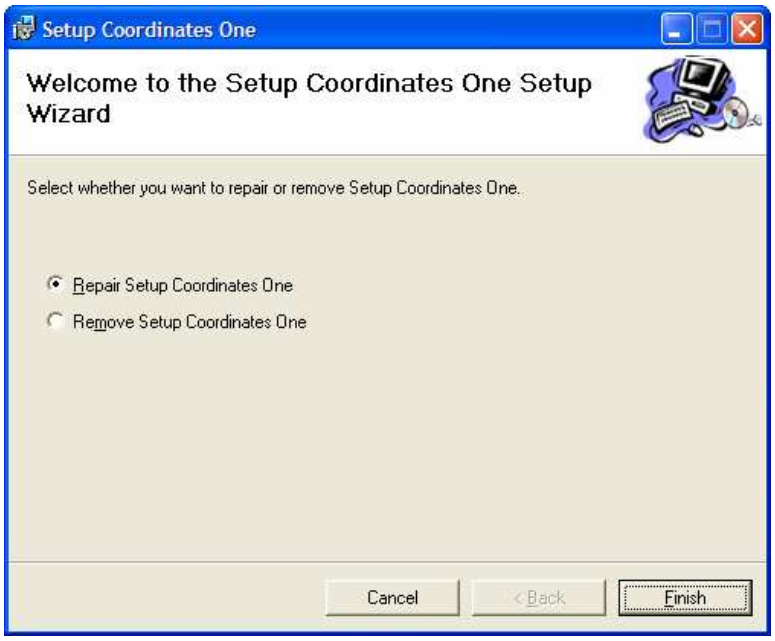
B.2 Requerimientos

Los requerimientos mínimos para ejecutar la aplicación son :
Microsoft Windows XP
Microsoft .NET Framework 2.0 o superior.
Espacio en disco duro: 350MB (incluyendo el modelo de elevación del terreno)

B.3 Instalación

Todos los archivos para la instalación de la aplicación se suministran en un único CD. Pulse el icono 

. El programa de instalación adjunto le asistirá durante el proceso.



La aplicación usa un modelo digital de elevaciones del territorio de Catalunya para mostrar el mapa del área a sobrevolar y calcular la elevación de un punto específico. Debido al gran tamaño del archivo (más de 300MB comprimidos) el fichero que contiene los datos se mantiene separado del código de la aplicación.
Para ejecutar la aplicación pulse el icono que aparecerá en su escritorio una vez finalizado el proceso de instalación.

Si todo el proceso se ha realizado con éxito la vista inicial de la aplicación será la que se adjunta en la figura B.2.

B.4 Descripción rápida

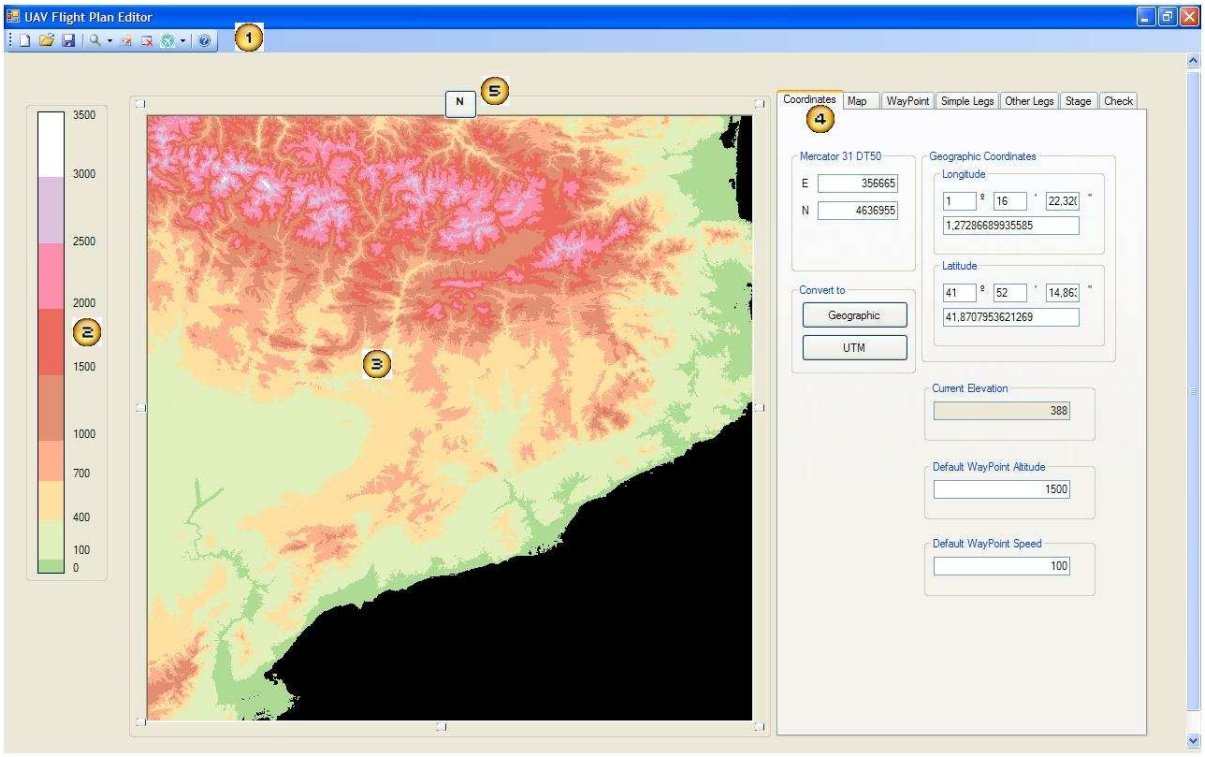


Fig. B.2 Aspecto aplicación

1. Leyenda
2. Mapa
3. Pestañas de funcionalidades
4. Barra de Herramientas
5. Botón norte de la herramienta pan

Desplace el cursor por encima de los controles para que aparezca sobre la pantalla una breve descripción de su funcionalidad.

B.5 Tab Control

Esta sección explica todos los menús y controles disponibles en el control de pestañas de la aplicación . Por simplicidad todos los controles que el operador necesita para la edición del plan de vuelo se encuentran en la parte derecha de la pantalla contenidos en un único control.

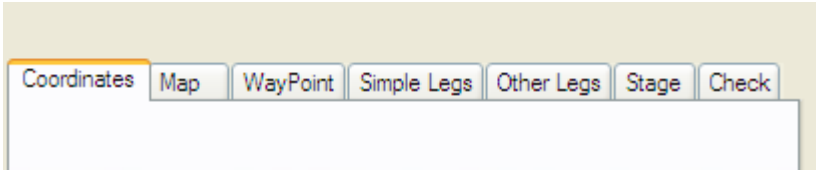


Fig. B.3. Pestañas

Las distintas pestañas disponibles y sus principales funciones son las siguientes :

- Coordinates: Pestaña que muestra las coordenadas del punto seleccionado tanto en coordenadas UTM como en coordenadas geográficas según la posición que ocupe el cursor sobre la imagen central.

- Map: Contiene un visor reducido del mapa donde se muestra la vista seleccionada y la escala utilizada por la vista principal.

- WayPoint: Permite añadir waypoints y editar sus características. También contienen un listado de todos los waypoints incluidos en el plan de vuelo que se está editando. Permite seleccionar un waypoint.

- Simple Leg: Permite crear legs a partir de dos waypoints previamente seleccionados y editar sus características

- Other Legs: Pestaña que permite editar los legs del tipo intersección, establecer que condición debe ser considerada al llegar a ellos y establecer el camino que se tomará por defecto.

- Stage: Permite agrupar y grabar el trabajo realizado hasta el momento en una fase del plan de vuelo.

- Check: Pestaña que permite verificar de una manera visual el plan de vuelo al completo y las referencias entre los objetos que lo componen sean consistentes. Permite al usuario detectar cualquier referencia errónea incluida en el plan de vuelo.

B.6.1 Pestaña Coordinates

Esta es la pestaña que aparece por defecto activada cuando se ejecuta por primera vez la aplicación. Esta pestaña informa de las coordenadas tanto UTM como geográficas que ocupa el cursor cuando se desplaza sobre el mapa. Algunos parámetros necesarios para la creación de un waypoint son recogidos aquí.

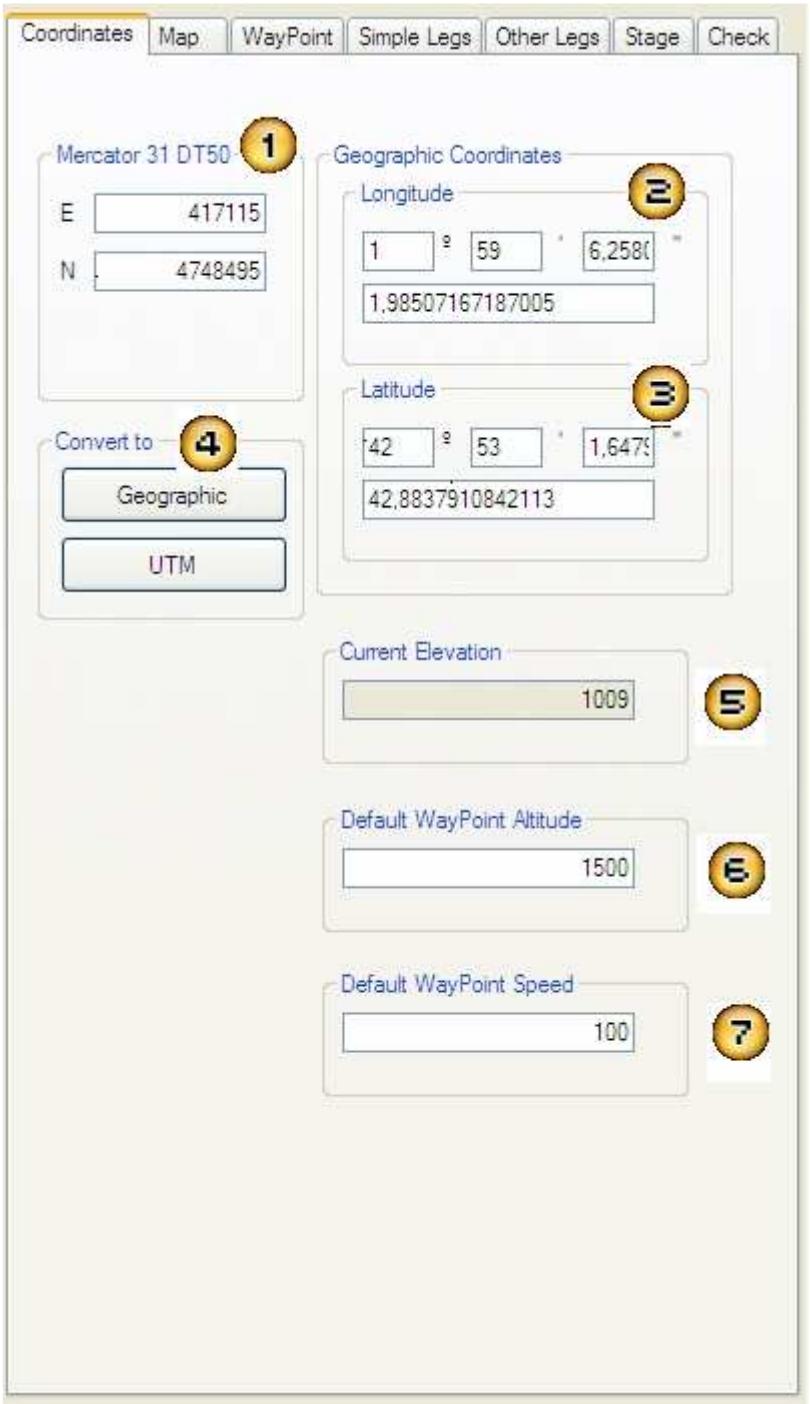


Fig. B.4 Pestaña Coordinates

Tabla B.1 Controles de la pestaña Coordinates

	Mercator 31DT50 Muestra las coordenadas UTM Easting y Northing de la posición que ocupa el cursor sobre el mapa.
	Geographic Coordinates Longitude *Longitud de las coordenadas geográficas* de la posición que ocupa el ratón sobre el mapa en grados, minutos y segundos. También se indican en grados en la parte inferior del control.
	Geographic Coordinates Latitude *Latitud de las coordenadas geográficas.*
	Convert to ... *Convertir a* ... Panel que permite la transformación del par de coordenadas UTM a coordenadas geográficas. Pulse Geographic para realizar este cambio y UTM para realizar el cambio inverso.
	Current Elevation. *Elevación actual*. Muestra la altura en metros de la posición actual del ratón sobre el mapa.
	Default WayPoint Altitude. *Altitud por defecto del waypoint*. Muestra el valor que se asigna al campo altitud al crear un nuevo waypoint. El valor asignado por defecto es de 1000 m.
	Default WayPoint Speed. *Velocidad por defecto del waypoint.* Muestra el valor que se asigna al campo velocidad al crear un nuevo waypoint. El valor asignado por defecto es de 100 Km/h.

B.6.2 Pestaña Map

Esta pestaña contiene los controles necesarios para activar la herramienta zoom y los parámetros de la herramienta pan. Muestra una vista en miniatura de todo el territorio e indica mediante un rectángulo sobre la misma el área ampliada. También la escala utilizada se muestra en esta pestaña.

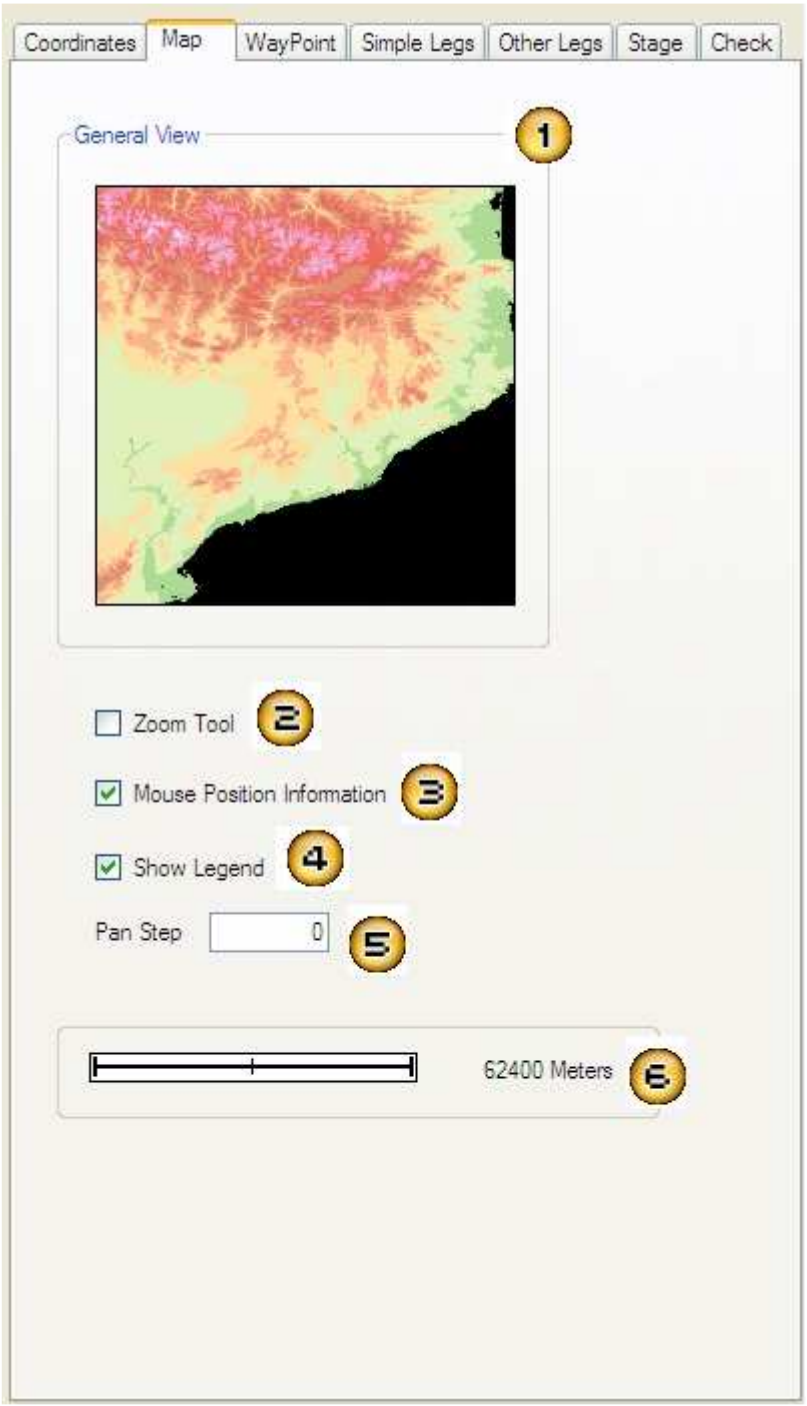


Fig. B.5 Pestaña Map

Tabla B.2 Controles de la pestaña Coordinates

	General View . *Vista general* . Muestra la vista inicial del modelo de elevación del terreno. Cada vez que el usuario cambia la vista haciendo uso de la herramienta zoom un rectángulo rojo indica la sección que ha sido ampliada sobre esta imagen. Si la imagen es pulsada la aplicación carga de nuevo la vista inicial.
	Zoom Tool . *Herramienta Zoom* . Activa la herramienta Zoom . La funcionalidad Mouse Position Information será desactivada mientras se hace uso de esta herramienta.
	Mouse Position Information. *Información sobre la posición del ratón*. Este checkbox hace que la aplicación regrese a su configuración inicial y vuelva a mostrar sobre el mapa las coordenadas que ocupa el ratón sobre el mapa. Moviendo el cursor sobre el mapa es posible ver en la pestaña Coordinates la posición que ocupa el ratón sobre el mismo.
	Show / i.e. Legenda *Oculta o muestra la leyenda*.
	Pan Stop. *Paso de pan*. Indica el paso que se aplica en los desplazamientos cuando los botones direccionales que rodean a la imagen central son pulsados. Aunque la aplicación calcula el paso adecuado de manera automática según el factor de zoom activo éste puede ser ajustado manualmente haciendo uso de esta caja de texto.
	Séale. *Escala*. Muestra la escala empleado para mostrar la vista actual.

B.6.3. Pestaña WayPoint

La pestaña waypoint contiene los controles para añadir o seleccionar un waypoint.

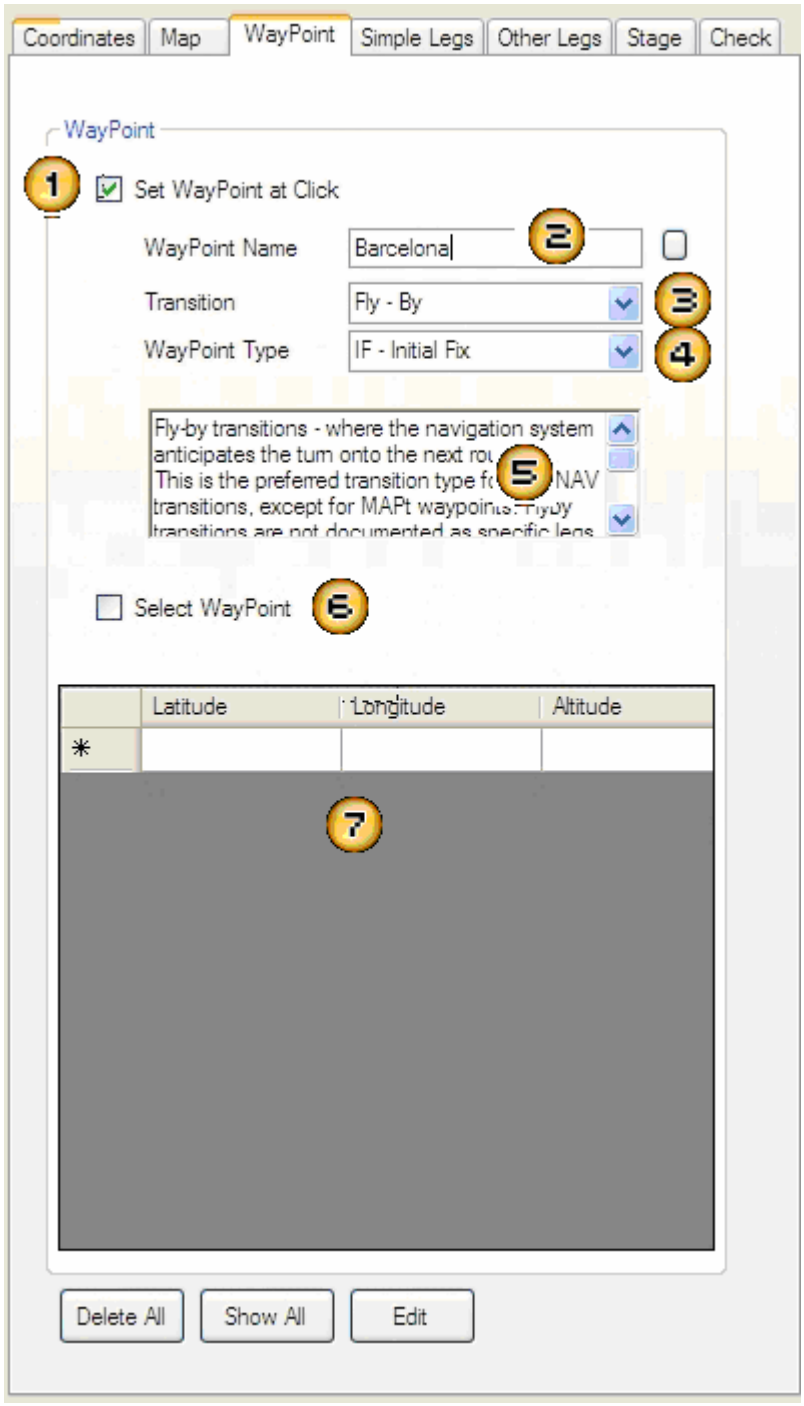


Fig. B.6 Pestaña WayPoint

Tabla B.3 Controles de la pestaña WayPoint

	NET WayPoint a.C. Clic . Cuando este checkbox está seleccionado la aplicación permite al operador añadir un nuevo waypoint en el plan de vuelo realizando un clic sobre la imagen central. El nuevo waypoint será mostrado marcado en rojo después de su introducción y sus coordenadas expresadas en grados aparecerán en [7].
	WayPoint Ñame . *Nombre del waypoint*. Esta caja de texto permite otorgar un nombre a cada waypoint incluido en el plan de vuelo.
	Transición. *Transición*. Permite al usuario indicar que tipo de transición debe realizar el UAV sobre el waypoint. Una breve explicación del tipo de transición seleccionado aparecerá en [5]
	WayPoint Tape. Permite al usuario seleccionar el tipo RNAV de waypoint.
	More Information. *Más información*. Muestra una breve descripción del tipo de transición seleccionado.
	Select Waypoint [Checkbox]. *Selection waypoint*. Permite al usuario seleccionar un waypoint introducido para su posterior edición . El waypoint seleccionado cambiará su color de rojo a amarillo.
	*Lista de WayPoints*. Ofrece una lista completa de los waypoints incluidos en el plan de vuelo. Un waypoint también puede ser seleccionado pulsando sus coordenadas sobre este control.

B.6.4. Pestaña Legs

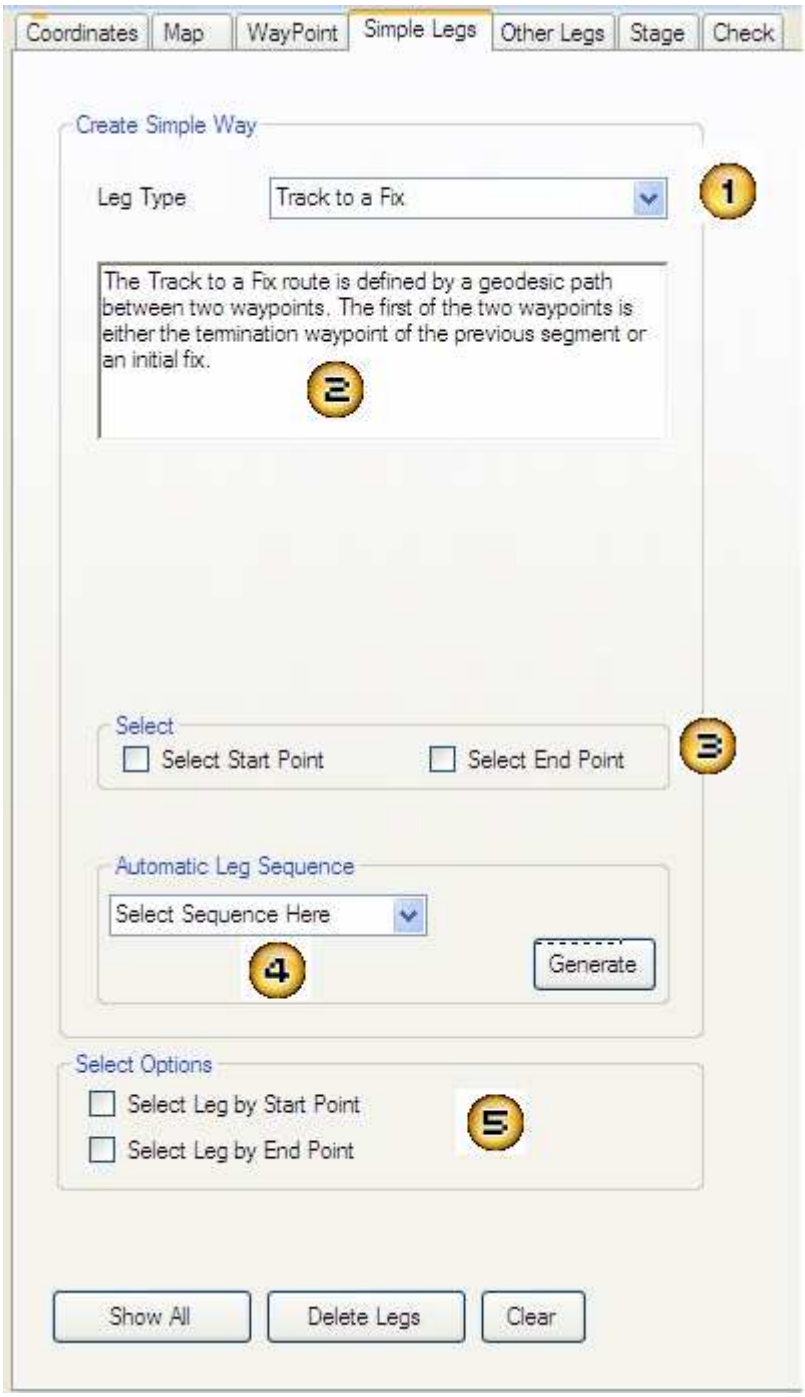


Fig. B.7 Pestaña Legs

Tabla B.4 Controles de la pestaña Legs

	Leg Tape . *Tipo de leg*. Esta caja de selección permite al operador seleccionar que tipo de leg se incluye en el plan de vuelo.
	Cuando un tipo de leg es seleccionado la definición del tipo y de sus parámetros aparecen en esta caja de texto. Si el leg necesita de parámetros adicionales para su definición los controles necesarios para su definición aparecerán debajo de esta caja. Véase apartado B.8.4
	Select Panel. *Panel de selección*. Permite al operador seleccionar el waypoint origen y destino para formar un nuevo leg. El primer waypoint se selecciona pulsando sobre el checkbox StartPoint (punto de origen) y seleccionando éste sobre el mapa. A continuación se pulsa sobre Select End Point y se selecciona sobre el mapa el waypoint de destino.
	Automatic Leg Sequence . *Secuencia de legs automática*. El botón Generate (Generar) permite generar secuencias de legs del mismo tipo y con los mismos parámetros de una manera rápida. Para más información sobre su uso consulte el apartado B.8.5.
	Select Options. *Opciones de selección*. Este par de checkbox permite al usuario elegir el modo en el que se selecciona un leg. Se puede seleccionar un leg por su waypoint origen o por su waypoint destino.

B.6.5. Pestaña Other Legs (Otros Legs)

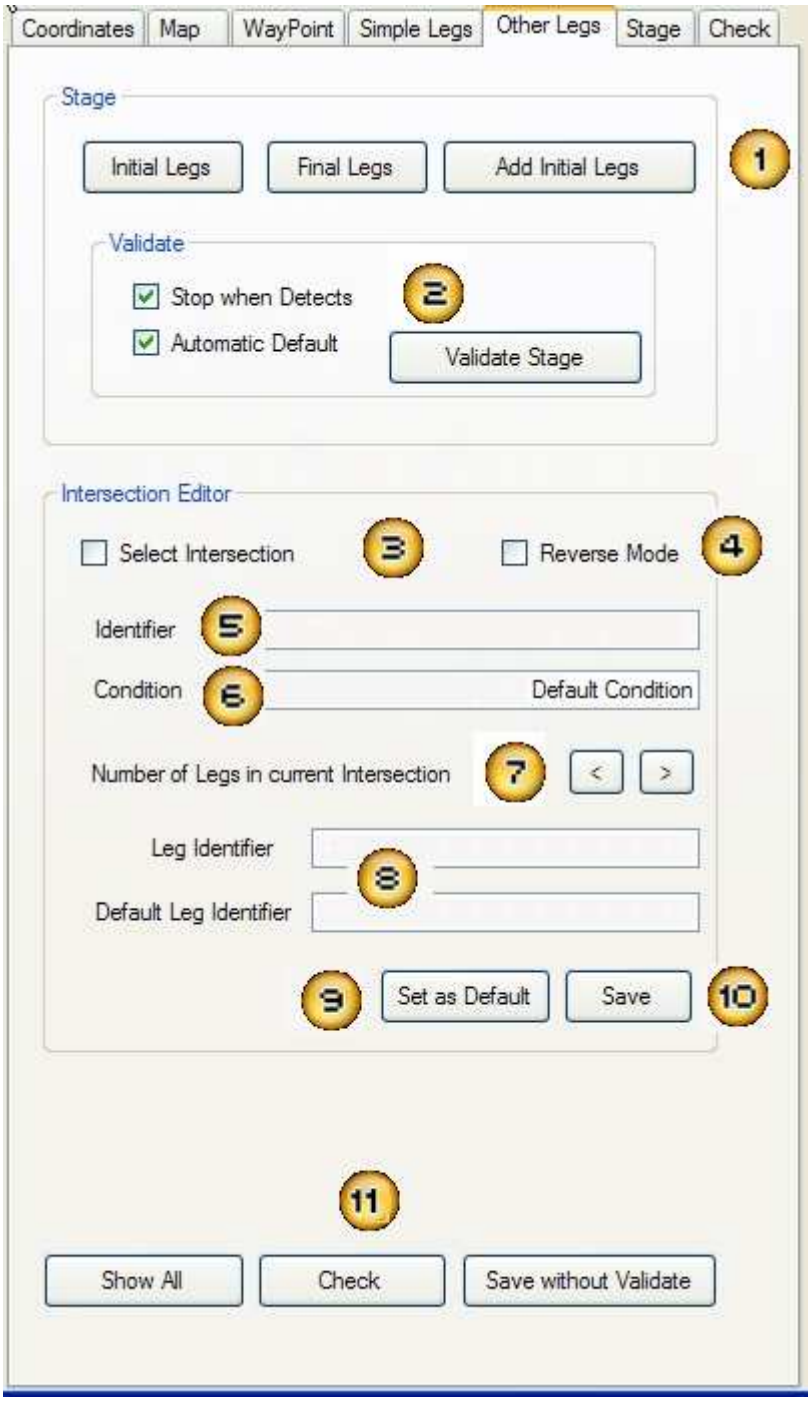


Fig. B.8 Pestaña Other Legs

Tabla B.5 Controles de la pestaña Other Legs

	Initial / Final Legs . Add Initial Legs. Esta serie de botones permiten visualizar los legs iniciales y finales que posee la fase en edición. El botón Add initial Legs extrae los legs finales de la fase anterior y los incluye como legs iniciales de la fase que se está editando.
	Validate Stage . Este botón inicia el proceso de validación de la fase actual. Véase el apartado B.8.9 para más información.
	Select Intersection. *Seleccionar Intersección*. Permite seleccionar las intersecciones con más de un posible leg siguiente para su edición.
	Reverse Mode. *Modo reverso*. Permite seleccionar las intersecciones con más de un leg previo para editarlas.
	Identifier. Muestra el identificador asignado al leg intersección.
	Condition. Campo para editar la condición ligada a la intersección.
	Botones para seleccionar el leg por defecto de la intersección. Cuando el usuario cambie el leg seleccionado la aplicación lo mostrará en color granate. El índice y número de legs disponibles se indican a la izquierda de estos botones.
	Leg Identifier / Default Leg Identifier . Muestra el identificador del leg que está seleccionado y el identificador del leg establecido por defecto.
	Set as Default. Pulsando este botón se establece como leg por defecto el leg seleccionado. Su identificador será mostrado en [8].
	Save. Graba los cambios realizados en la intersección.
	Show All / Check / Save without Validate

B.6.6. Pestaña Stage (Fases)

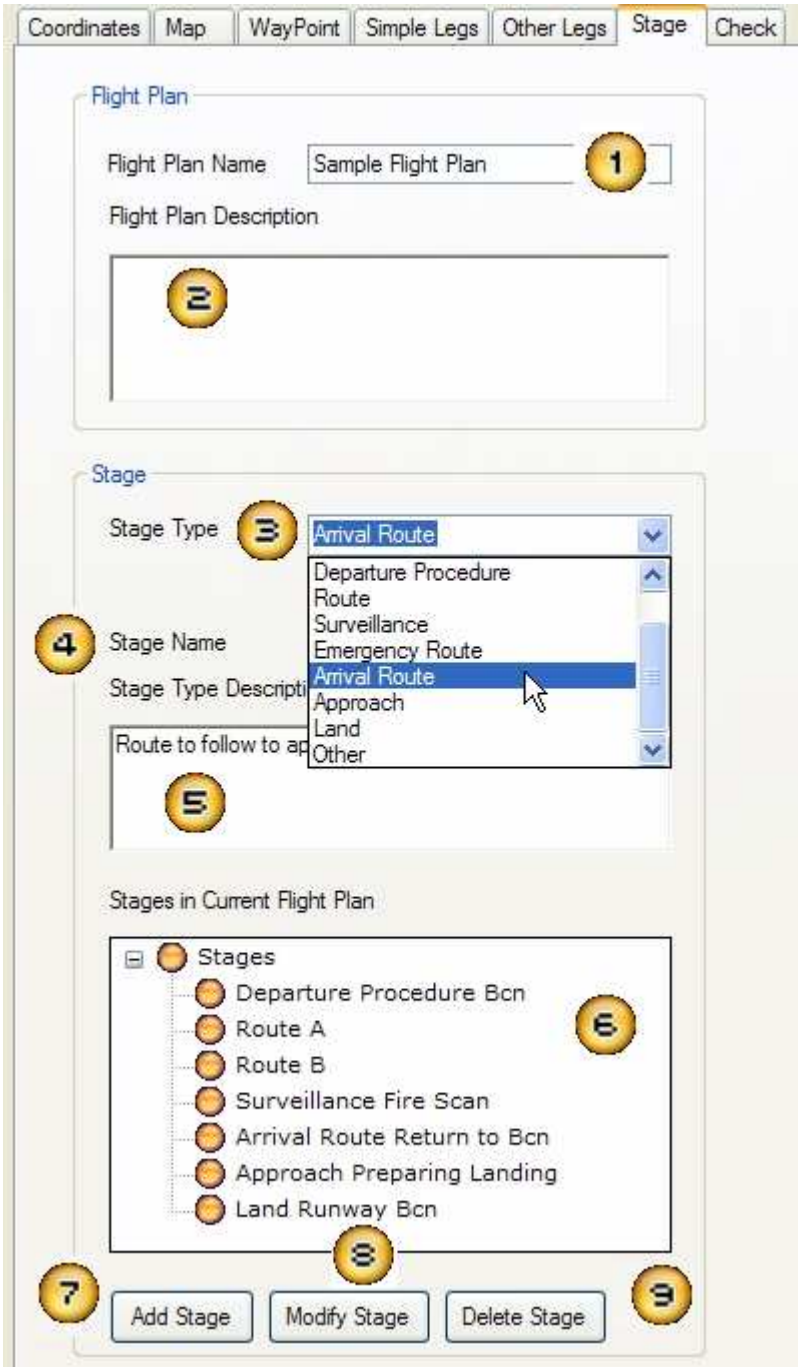


Fig. B.9 Pestaña Stage

Tabla B.6 Controles de la pestaña Stage

	Flight Plan Name. Campo que asigna un nombre al plan de vuelo.
	Flight Plan Description. *Descripción del plan de vuelo*. Almacena una breve descripción para el plan de vuelo que se está editando.
	Stage Type. *Tipo de fase*. Permite seleccionar el tipo de fase de entre los tipos. Al cambiar el tipo de fase una breve descripción de la misma aparecerá en Stage Type Description [5].
	Stage Name. *Nombre de la fase*. Campo utilizado para asignar un nombre a la fase.
	Stage Type Description. *Descripción del tipo de fase*. Muestra una breve descripción del tipo de fase seleccionado.
	Stages in Current Flight Plan. *Fases del plan de vuelo actual*. Muestra todas las fases incorporadas al plan de vuelo. Si se pulsa el nombre de una fase la aplicación mostrará la fase seleccionada sobre el mapa.
	Add Stage. *Añadir fase*. Almacena el trabajo actual como una nueva fase del plan de vuelo.
	Modify Stage. *Modificar fase*. Almacena los cambios realizados en la fase seleccionada en [6].
	Delete Stage. Borra la fase seleccionada en [6] del plan de vuelo.

B.6.7. Pestaña Check . Verificar

Fig. B.10 Pestaña Check

Tabla B.7 Controles de la pestaña Check

	Reverse Mode [CheckBox]. *Modo reverso*. Activa el modo reverso de verificación. En este modo las referencias que se verifican empiezan en los legs finales de la aplicación y la visualización resigue las referencias al leg previo de cada leg contenido en la fase.
	Stop at Final Legs. *Detenerse en los legs finales*. Si este checkbox está activado la visualización se detendrá al llegar al final del recorrido para cada leg inicial posible.
	Interval. *Intervalo*. Indica a la aplicación cada cuantos milisegundos debe mostrar el siguiente leg.
	Stage. Muestra el nombre de la fase que se está verificando.
	Identifier. *Identificador*. Muestra el identificador de la intersección.
	Initial Way X of Y. *Camino inicial X de Y*. Muestra el índice del leg inicial que se ha tomado para iniciar la visualización (X) y el número total de legs iniciales de los que dispone la fase (Y). De manera análoga muestra los legs finales cuando el modo reverso [1] está activado.
	Start. *Comenzar*. Botón que inicia la visualización de la animación.
	Stop. *Detener*. Detiene la animación hasta que Continue [9] o Start [7] son pulsados.
	Continue. *Continuar*. Reemprende la visualización.

B.6 Otros menús y Controles

B.7.1 Barra de Herramientas

Aunque la barra de herramientas puede ser desplazada al iniciar la aplicación ésta aparece en la parte superior izquierda de la pantalla. Su aspecto se muestra en la figura B.11.

Fig. B.11 Barra de Herramientas

La barra de herramientas da acceso rápido a las siguientes funciones:

-  Crear un plan de vuelo nuevo.
-  Abre un cuadro de diálogo que permite abrir un fichero XML previamente almacenado.
-  Abre un cuadro de diálogo que permite grabar un plan de vuelo en formato XML.
-  Activa la herramienta zoom de la aplicación.
-  Acceso rápido para añadir un nuevo waypoint al plan de vuelo.
-  Elimina todos los waypoints incluidos en el plan de vuelo.
-  Da acceso al formulario de visualización.
-  Accede a la ayuda de la aplicación.

B.7.2 Funcionalidad botón derecho sobre la imagen del mapa

Al pulsar el botón derecho sobre el mapa de la aplicación aparece el menú contextual que aparece en la figura B.12.

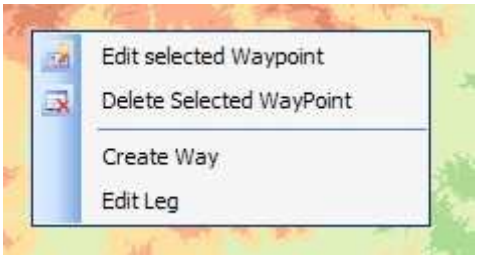


Fig. B.12 Menú Imagen Central

B.7 ¿ Como puedo ...?

B.8.1.Añadir un waypoint

Antes de añadir un waypoint verifique que los valores asignados por defecto para la creación de un waypoint de altitud y velocidad son los adecuados en la pestaña Coordinates.

Active el checkbox AddWayPoint at Click en la pestaña WayPoint y cumplimente el resto de parámetros del mismo (nombre y tipo de transición). Una breve descripción del tipo de transición seleccionado aparecerá debajo de su selección.

Si intenta incluir un waypoint con una altitud asignada por defecto inferior a la altitud de la elevación en el punto clickado la aplicación mostrará el mensaje de advertencia de la figura B.13.

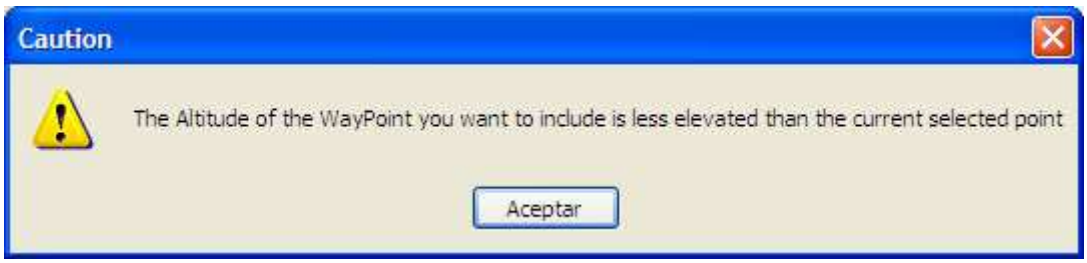


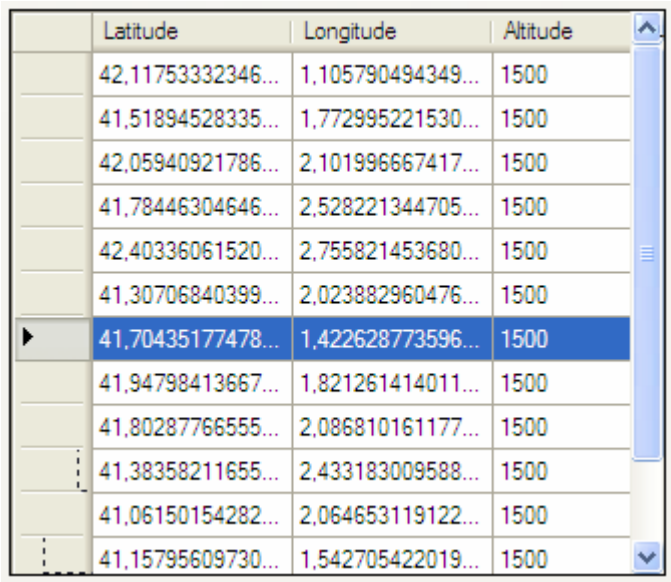
Fig. B.13 Mensaje de advertencia

B.8.2 Seleccionar un waypoint

Un waypoint previamente incluido en el plan de vuelo puede ser seleccionado de dos maneras distintas.

1. Activando el checkbox Select waypoint presente en la pestaña waypoint y seleccionando el mismo sobre el mapa.

2. Pulsando sus coordenadas en el control Lista de waypoints presente en la misma pestaña. Este control se muestra en la figura B.14



	Latitude	Longitude	Altitude
	42,11753332346...	1,105790494349...	1500
	41,51894528335...	1,772995221530...	1500
	42,05940921786...	2,101996667417...	1500
	41,78446304646...	2,528221344705...	1500
	42,40336061520...	2,755821453680...	1500
	41,30706840399...	2,023882960476...	1500
▶	41,70435177478...	1,422628773596...	1500
	41,94798413667...	1,821261414011...	1500
	41,80287766555...	2,086810161177...	1500
	41,38358211655...	2,433183009588...	1500
	41,06150154282...	2,064653119122...	1500
	41,15795609730...	1,542705422019...	1500

Fig. B.14 Lista de waypoints

Cuando un waypoint sea seleccionado éste se mostrará en color amarillo en vez de en color rojo como es habitual.

B.8.3 Eliminar un waypoint

Seleccione el waypoint que desea eliminar y haga click sobre el botón derecho del ratón cuando éste se encuentre sobre el mapa. Seguidamente seleccione la opción Delete WayPoint del menú contextual mostrado en la figura B.15.

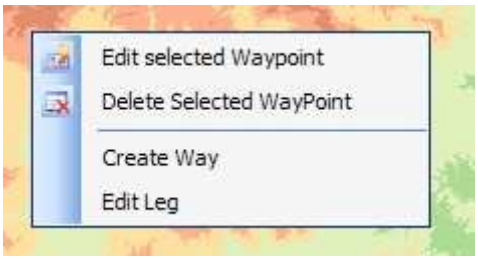


Fig. B.15 Eliminar un waypoint

B.8.4 Crear un nuevo leg

Todos los controles necesarios para la creación de un nuevo leg se encuentran agrupados en el grupo de controles Create Simple Way.

1. Seleccionar el origen
2. Seleccionar el destino
3. Seleccionar el tipo de leg que desea crear.
4. Introducir el resto de parámetros que definen al leg.

El cuadro de diálogo se muestra en la figura B.16

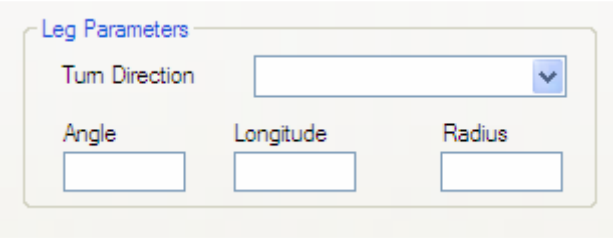


Fig. B.16 Cuadro de dialogo parámetros leg

Estos parámetros pueden variar según el leg que se esté creando. En el caso de un leg Radius to fix la trayectoria queda definida por su centro y los parámetros Radius (radio) y TurnDirection (sentido de giro). Entendiendo Radius como el radio del círculo que el avión debe trazar entre los dos waypoints y con centro sobre la mediatriz del segmento que los une.

En el caso de un leg Hold Pattern la trayectoria se define con el sentido de giro, el ángulo que define los cuadrantes de aproximación al waypoint y las longitudes del segmento recto y circular del patrón.

5. Pulse el botón derecho del ratón cuando éste esté sobre el mapa y seleccione la opción Create Way del menú contextual.

La figura B.17 muestra el aspecto de esta pestaña al terminar el proceso de creación de un leg del tipo Hold Pattern con sentido de giro en sentido horario, un ángulo de 30 grados, 1000 metros de longitud del segmento recto del patrón y 200 metros de radio del segmento circular.

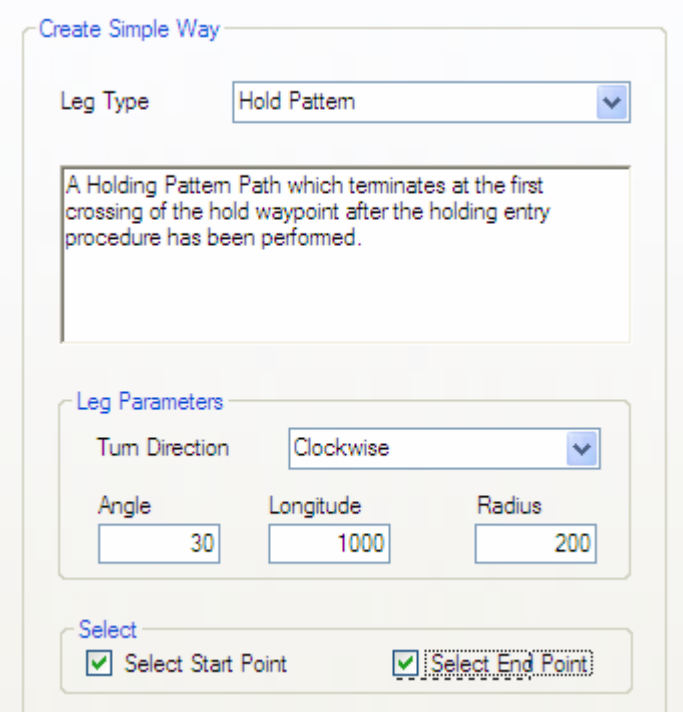


Fig. B.17 Ejemplo creación leg

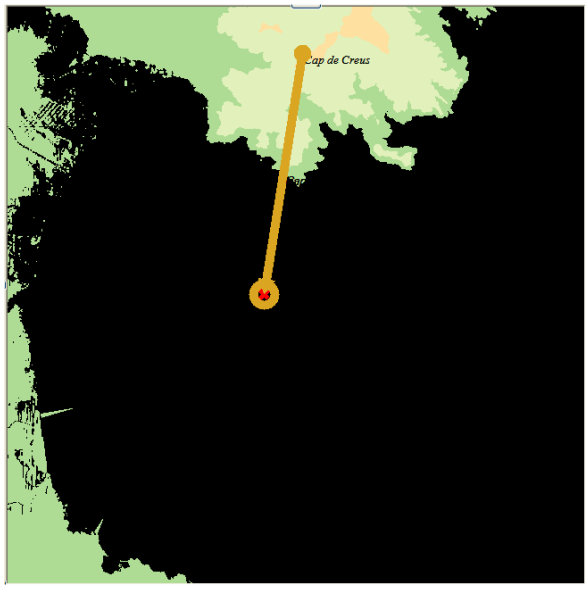


Fig. B.18 Visualización Leg Hold Pattern

B.8.5 Generar series de legs

Los controles que le ayudarán a crear series de legs de una manera rápida están presentes en la pestaña Legs y recogidas en el grupo Automatic Leg Sequence. Fig. B.19.

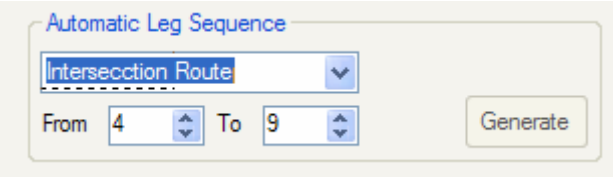


Fig. B.19 Automatic Leg Sequence

Los legs generados en secuencia deben ser del mismo tipo y tener todos los mismos parámetros.

Deberá primero crear los waypoints que formarán parte de la secuencia. Luego seleccione el tipo de secuencia que desea crear.

Las opciones de las que dispone son:

- From Origin (desde el origen) : Creará una secuencia del mismo tipo de leg con todos los waypoints introducidos en la fase actual.

- From Last (desde el útimo) : Creará una secuencia de legs empezando desde el último leg perteneciente a la última secuencia creada hasta el último waypoint introducido posteriormente.

- Intersection Route . Creará un nuevo camino que corte con los presentes en la fase creando una derivación desde el waypoint que seleccione en el campo From hasta el waypoint seleccionado en el campo To y continuará la secuencia hasta el último waypoint introducido. La aplicación mostrará el puente mientras está seleccionado los waypoints que lo componen.

Cuando la visualización por pantalla se corresponda con la secuencia que desea crear pulse el botón Generate (generar).

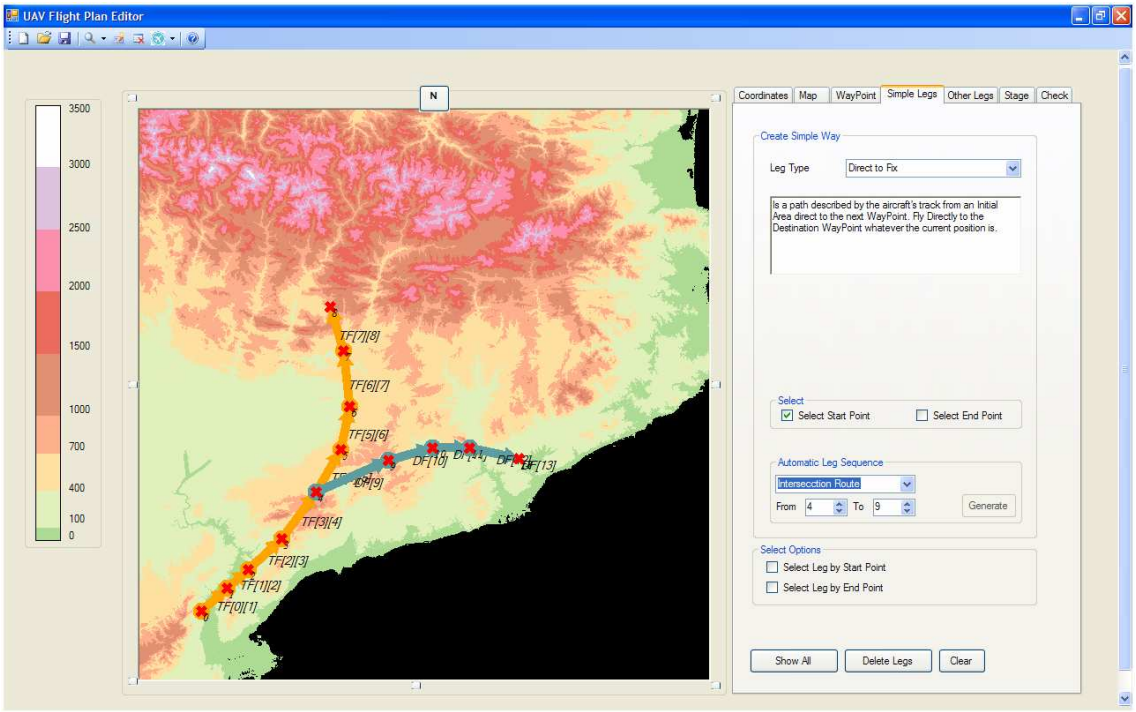


Fig. B.20 Ejemplo secuencia de legs

La figura B.20 muestra el aspecto que ofrece la aplicación tras crear una secuencia de legs Track to fix utilizando la opción From Origin y posteriormente crear una secuencia de legs con la opción Intersection Route desde el waypoint número 4 como From al waypoint 9 como To.

B.8.6 Seleccionar un leg existente

Un leg puede ser seleccionado de dos maneras distintas:

1. Según el waypoint de origen. El leg seleccionado aparecerá iluminado en color amarillo indistintamente del color de representación habitual.

2. Según el waypoint de destino

Si la aplicación detecta más de un leg con el mismo punto de origen iluminará ambos legs en color amarillo. Esta situación se refleja en la Fig B.21

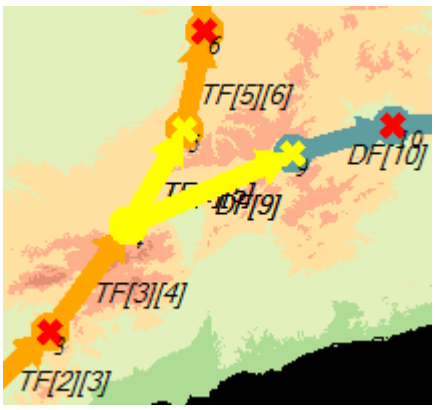


Fig. B.21 Seleccionar leg

Para seleccionar un leg del tipo intersección utilice los checkbox SelectIntersection o Reverse Mode presentes en la pestaña Other Legs.

B.8.7 Establecer legs iniciales y legs finales de una fase

La aplicación detectará automáticamente los legs iniciales y finales de cada fase al validar la misma. Véase el apartado B.8.9 para más información sobre la validación de una fase. Si desea verificar esta asignación pulse los botones Initial Legs o Final Legs presentes en la pestaña Stage.

La figura B.22 muestra el aspecto de un leg inicial cuando es mostrada por la aplicación.

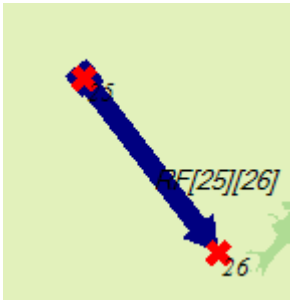


Fig. B.22 Leg Inicial

La figura B.23 muestra el aspecto de un leg final cuando es mostrado por la aplicación.

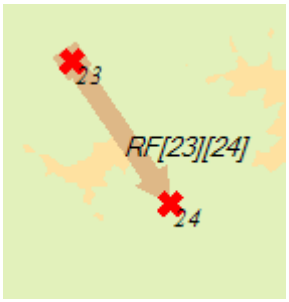


Fig. B.23 Leg Final

El botón Add Initial Legs extrae los legs finales de la fase anterior y los incluye como legs iniciales de la fase que se está editando.

B.8.8 Editar una intersección

La aplicación detectará automáticamente los legs intersección requeridos al pulsar sobre el botón Validate Stage presente en la pestaña Other Legs. Una vez realizado este proceso el usuario puede editar las intersecciónes.

Todos los controles necesarios para su edición se encuentran agrupados en el grupo Intersection Editor que se muestran en la figura B.24.

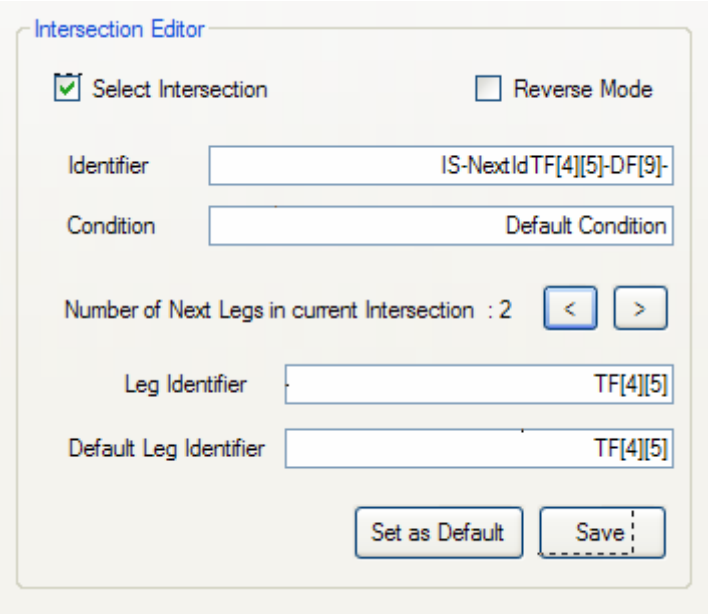


Fig. B.24 Cuadro de edición leg intersección

Las intersecciones son iluminadas en color violeta y el leg asignado por defecto se muestra en color verde. La figura B.25 muestra como la aplicación muestra este leg.

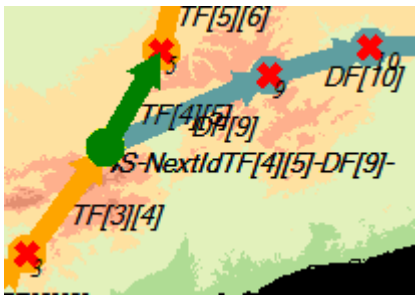
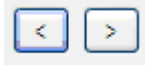


Fig. B.25 Aspecto leg default

La condición que se establece para la intersección puede ser editada en la caja de texto llamada Condition. El valor por defecto de este campo es “Default Condition” que indica que el leg establecido por defecto seguirá siempre a la intersección.

Para modificar el leg asignado por defecto utilice los botones 

. Los posibles legs por defecto a asignar aparecerán en color granate. La figura B.26 muestra la misma intersección con sus dos posibles legs siguientes antes de ser seleccionados. Al pulsar sobre los botones el aspecto de la intersección cambiará entre una imagen y otra.

Fig. B.26 Leg Default

Cuando el leg que quiera establecer como default este iluminado pulse Set as Default.

B.8.9 Validar una fase

Consiste en un proceso automático de verificación que realiza la aplicación para asignar las referencias entre los legs. Establecer para cada leg cual será el leg siguiente y cual su previo.

Este proceso, con el fin de ayudar al usuario se detiene tras detectar un leg inicial, un leg final o una intersección:

- Detecta los legs que no tienen ningún posible leg previo y los asigna como legs iniciales de la fase actual.

- Detecta los legs que no tienen ningún posible leg siguiente y los asigna como legs finales de la fase actual.

La aplicación mostrará los siguientes mensajes cuando detecte un leg inicial o un leg final.

Fig. B.27 Mensajes de validación para los legs iniciales y finales.

- Detecta los caminos que se cortan y crea el leg intersección. La aplicación distingue entre intersecciones con más de un leg siguiente e intersecciones con más de un leg previo. La figura B.28 muestra el mensaje que aparece al detectar una intersección con más de un leg siguiente.

Fig. B.28 Mensaje de validación para una intersección

Los legs que se asignaran como previos aparecen iluminados en azul. El leg al cual se están asignando las referencias aparece iluminado en verde y los legs que se asignaran como siguientes aparecen en rojo. De detectar más de un posible leg siguiente la aplicación creará una intersección.

La figura B.29 ilustra este proceso.

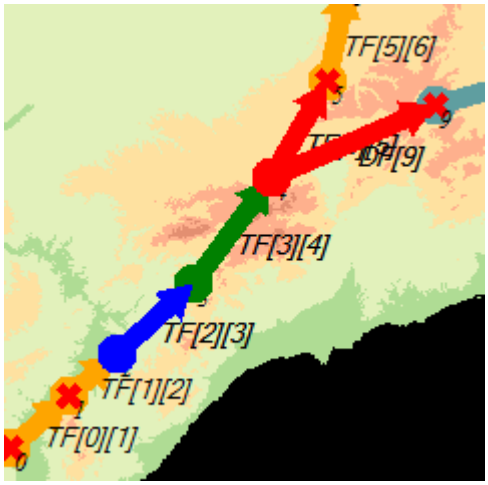


Fig. B.29 Proceso de validación

Si desea que la aplicación realice este proceso sin detenerse en ningún momento el checkbox Stop When Detects debe estar desactivado.

B.8.10 Añadir una nueva fase del plan de vuelo

Cuando las referencias entre los legs han sido establecidas y validadas es posible grabar el trabajo actual como una nueva fase del plan de vuelo. Consulte B.8.9 para más información al respecto. Si se intenta grabar la fase sin haberla validado la aplicación mostrara el siguiente mensaje.

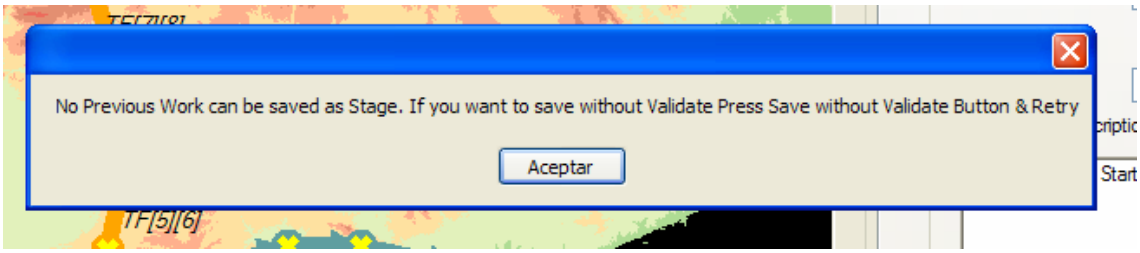


Fig. B.30

1. Seleccionar el tipo de fase. Cuando el tipo de fase es modificado aparece una breve descripción de la misma en la caja de texto Stage Type Description.

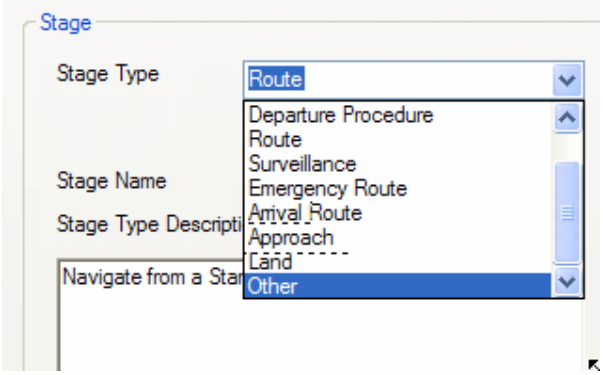


Fig. B.31 Nombre y selección del tipo de fase

2. Asigne un nombre a la fase. La figura B.32 muestra como ejemplo una fase con el nombre Route to Bcn del tipo Route.

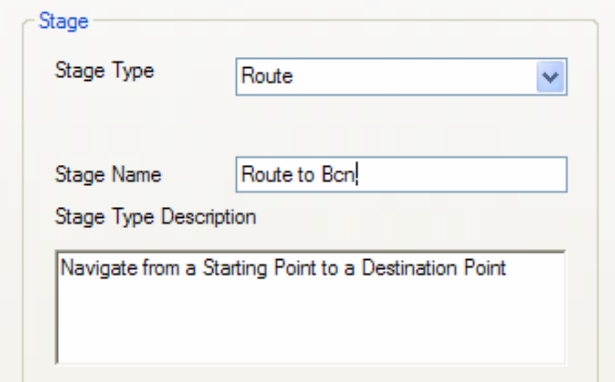


Fig. B.32 Asignar tipo y nombre a una fase

Los leg editados hasta el momento son incluidos como una nueva fase del plan de vuelo en el control Stages in Current Flight Plan. La fase incluida se muestra en el control como la concatenación del tipo de fase y el nombre asignado a la misma. La figura B.33 muestra el aspecto de este control con siete fases del plan de vuelo incluidas.

Fig. B.33 Fases incluidas en el plan de vuelo

B.8.11 Cargar una fase

Para cargar una fase contenida en el plan de vuelo para su visualización o modificación pulse sobre el nombre de la fase en el control Stages in Current Flight Plan presente en la pestaña Stage y mostrado en la figura B.33.

B.8.12 Reutilizar una fase previamente almacenada

Para reutilizar componentes de una fase previamente utilizada cargue la fase como se indica en el apartado B.8.11 y realice los cambios pertinentes en la misma. Grabe las modificaciones con un nuevo nombre siguiendo las indicaciones ofrecidas en el apartado B.8.10.

B.8.13 Cargar un plan de vuelo

El botón que permite cargar un plan de vuelo previamente almacenado en formato XML es accesible desde la barra de herramientas. Pulse  y el siguiente cuadro de diálogo se mostrará para que pueda elegir el fichero XML que desea cargar.

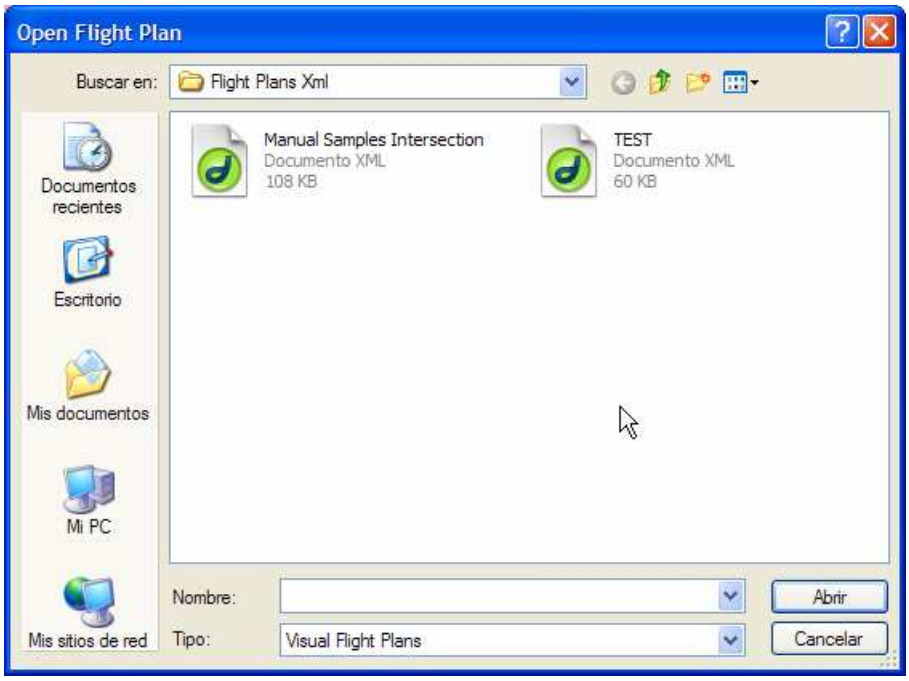


Fig. B.34 Abrir un plan de vuelo

B.8.14 Grabar un plan de vuelo

Primero añada un nombre y una descripción al plan de vuelo antes de grabar.

Pulse grabar en la barra de herramientas. Esta función es accesible desde la barra de herramientas mediante el botón  .

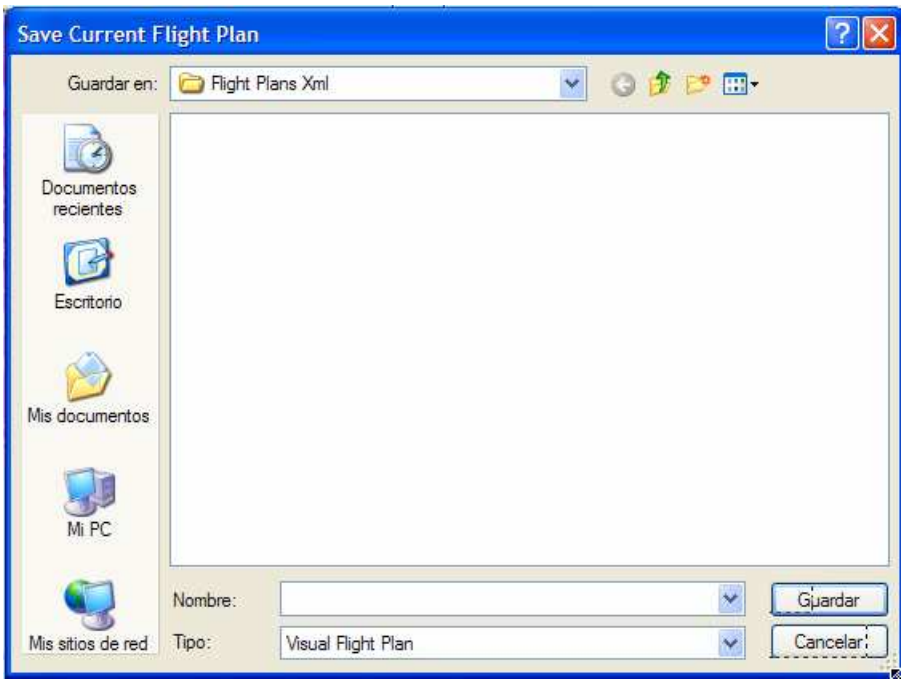


Fig. B.35 Grabar un plan de vuelo

Cuando el plan de vuelo se graba la aplicación genera el fichero XML que lo representa. El cuadro de diálogo de la figura B.35 se mostrará. Seleccione el directorio donde desea grabar el plan de vuelo e introduzca el nombre que desea asignar al fichero si este es distinto del asignado en el campo Flight Plan Name de la pestaña Stage. Pulse el botón Guardar para finalizar.

B.8.15 Validar una plan de vuelo

Para poder validar un plan de vuelo al menos una fase debe haber sido validada y las relaciones entre los legs deben haber sido establecidas.

Si intenta visualizar estas relaciones antes de validar una fase la aplicación mostrará el siguiente mensaje :

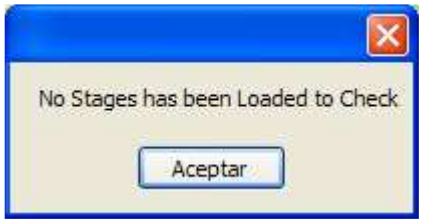


Fig. B.36 Mensaje Check

Para verificar que las relaciones establecidas entre los legs de una determinada fase sean correctas seleccione la pestaña Check y pulse el botón Start.

La aplicación empezará a iluminar en color verde los legs empezando por cada uno de los legs establecidos como iniciales y pasará al siguiente leg según el intervalo que especifique en la caja de texto interval.

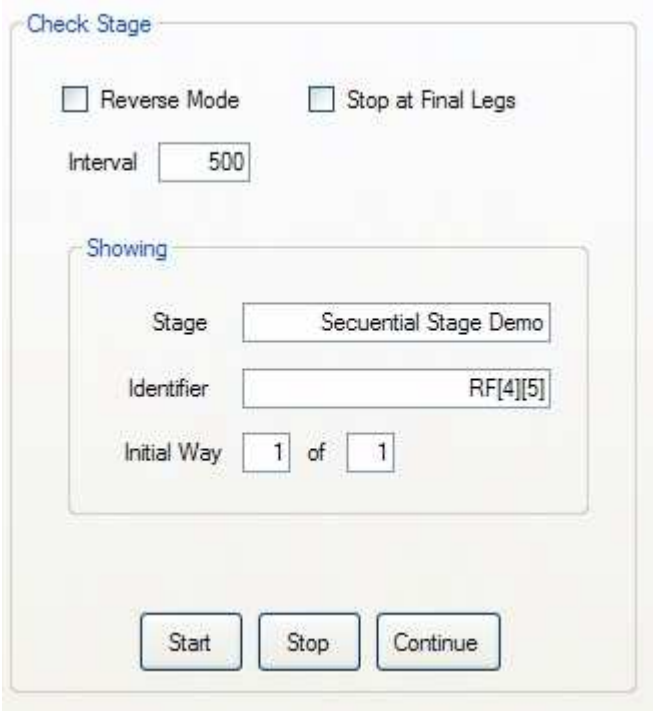


Fig. B.37

Utilice el valor de la caja de texto Interval para indicar a la aplicación cuantos milisegundos debe esperar antes de proceder a iluminar el siguiente leg.

Si desea verificar las relaciones en modo reverso active el checkbox Reverse Mode presente en la pestaña check.

En el grupo de controles Showing muestra:

- El nombre de la fase que está verificando.
- El identificador del leg que está iluminando.
- En el campo Initial Way muestra el valor del leg inicial donde ha comenzado la animación y el número total de legs iniciales de los que consta la fase.

Con el checkbox Stop at Final Legs activado la aplicación se detiene al llegar a cada leg final de la fase. De otra manera la aplicación continúa con la animación desde el leg inicial de la fase siguiente.

Utilice el botón de Start para iniciar el proceso de verificación. El botón de Stop detiene la animación hasta que el botón Continue es pulsado. Si desea iniciar la verificación en cualquier momento pulse de nuevo el botón de Start. La figura muestra el momento en que la aplicación ilumina el leg RF[4][5] de la fase.

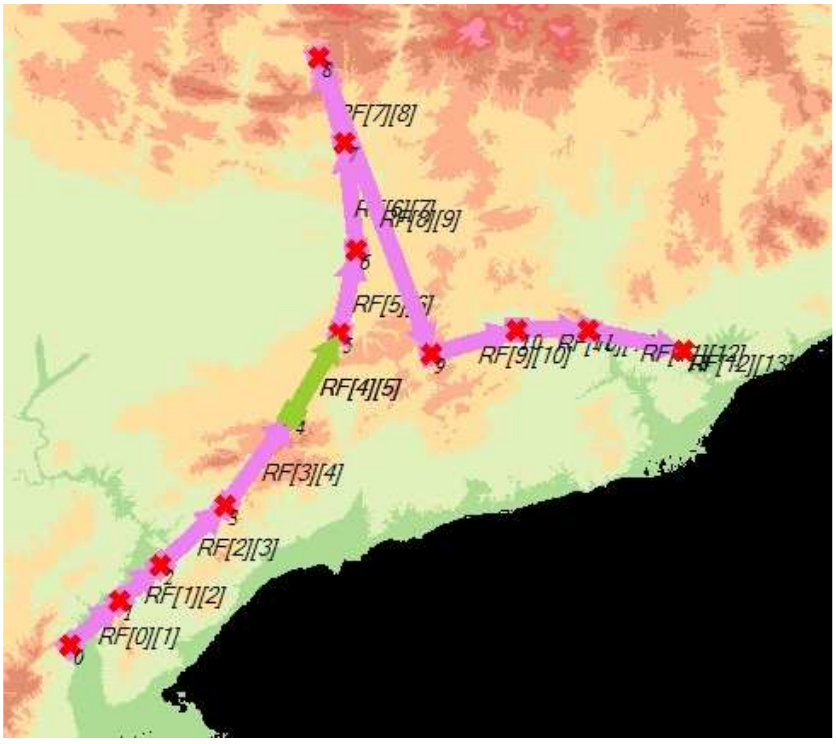


Fig. B.38 Proceso de validación